

ORSAY

Série A

N° 949

T H E S E S

PRESENTEES

**A L'UNIVERSITE DE PARIS-SUD
CENTRE D'ORSAY**

POUR OBTENIR

LE GRADE DE DOCTEUR ES SCIENCES PHYSIQUES

par

Francis ABOULKER

PREMIERE THESE

**ETUDE DU PROJET D'UN CALCULATEUR MODULAIRE
TRAVAILLANT EN MULTITRAITEMENT**

DEUXIEME THESE

Propositions données par l'Université

Soutenues le 12 mai 1972, devant la Commission d'Examen

MM. DEBRAINE

Président

**BLAQUIERE
DONADIEU**

Examineurs

Mlle BESTOUGEFF

ORSAY

Série A

N° 949

T H E S E S

PRESENTEES

**A L'UNIVERSITE DE PARIS-SUD
CENTRE D'ORSAY**

POUR OBTENIR

LE GRADE DE DOCTEUR ES SCIENCES PHYSIQUES

par

Francis ABOULKER

PREMIERE THESE

**ETUDE DU PROJET D'UN CALCULATEUR MODULAIRE
TRAVAILLANT EN MULTITRAITEMENT**

DEUXIEME THESE

Propositions données par l'Université

Soutenues le 12 mai 1972, devant la Commission d'Examen

MM. DEBRAINE

Président

BLAQUIERE

DONADIEU

Examineurs

Mlle

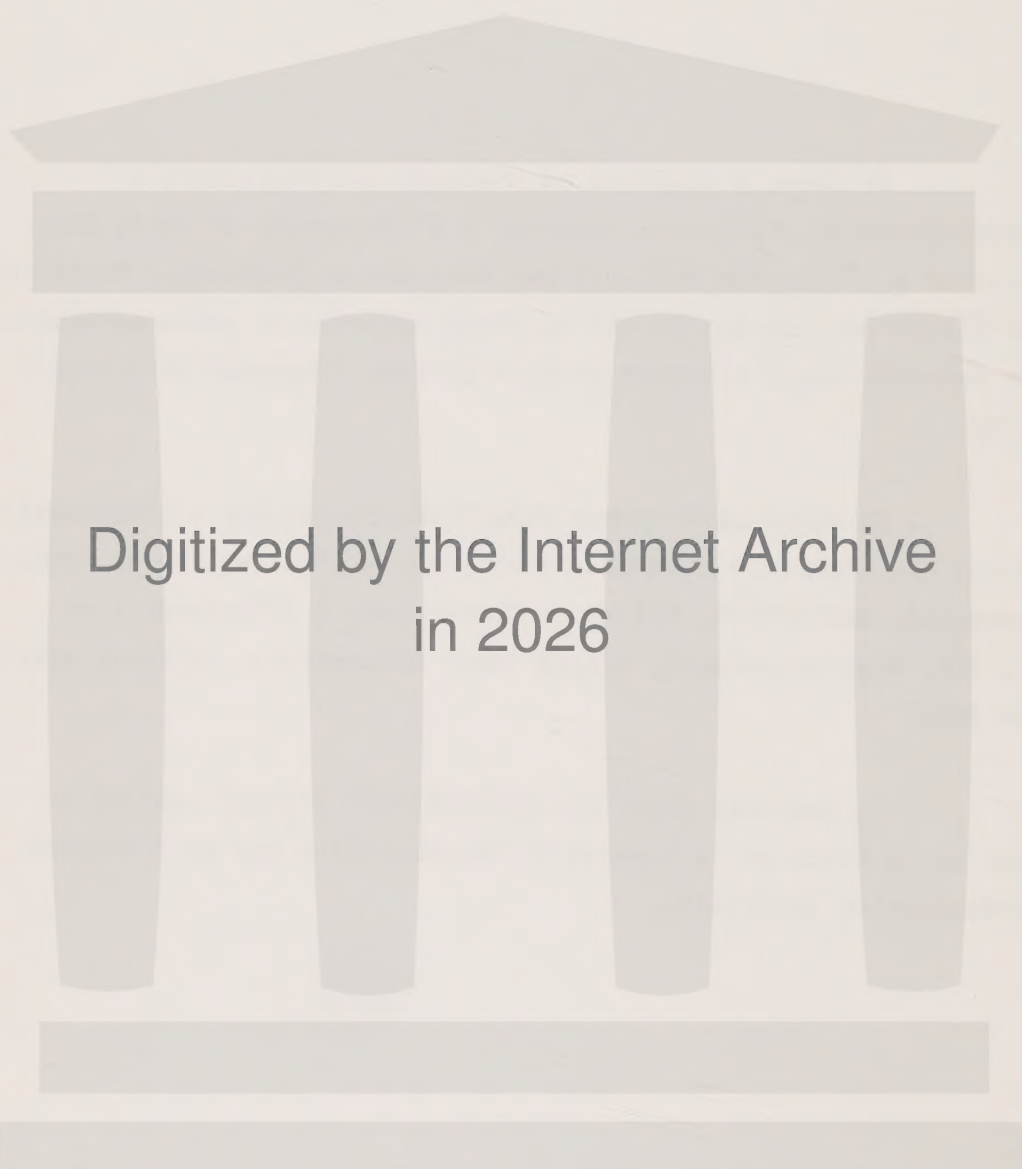
BESTOUGEFF



Je tiens à exprimer ma profonde reconnaissance à Monsieur DEBRAINE, Professeur Associé à l'Université de Paris Sud, Professeur à l'Institut National des Sciences et Techniques Nucléaires, qui a dirigé mes travaux et m'a prodigué conseils et encouragements. Je le remercie aussi vivement de m'avoir fait l'honneur de présider ce jury.

Je remercie Monsieur BLAQUIERE, Professeur à l'Université de Paris VII, Monsieur DONADIEU, Professeur à l'Université de Paris Sud, Mademoiselle BESTOUGEFF, Maître de Conférences à l'Université de Paris VII, d'avoir accepté, malgré leurs occupations, de faire partie de ce jury.

Je remercie aussi Mme AIGUESPARSES d'avoir réalisé la frappe de ce texte et le Service de Documentation qui en a réalisé la présentation matérielle.



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41547106_0021

CHAPITRE I

INTRODUCTION

Depuis les premières machines à tubes ou à relais, la technologie des calculateurs a fait des progrès immenses au point d'atteindre les limites imposées par les lois de la physique dans le domaine de la propagation des signaux électriques. Ce progrès a été principalement l'oeuvre des constructeurs et a été caractérisé par l'annonce périodique de gammes nouvelles de machines avec une architecture quelquefois radicalement différente de celle des séries précédentes. Ce progrès a été accompagné par la création ex nihilo de grands systèmes de programmation complexes.

Ce mode de développement a produit de nombreux cas d'indigestion des systèmes et ne peut continuer sur le même rythme car l'accroissement rapide du nombre de systèmes installés, dans des domaines aussi vastes que ceux du temps réel ou de la simulation, par exemple, a conduit à un investissement accru dans leur élaboration. L'investissement total en équipements, programmes, et, en particulier, dans la création de fichiers et la formation, est maintenant tel que, aussi bien pour les fabricants que pour les usagers, le progrès à venir sera une évolution plutôt qu'une révolution périodique. Ce changement dans l'orientation de la politique ne doit cependant pas tendre à maintenir le statu quo. Le progrès à venir sera donc plus tourné vers une évolution des outils de calcul actuels qui forment une base satisfaisante.

Le développement futur dans le domaine de la circuiterie demandera un meilleur rapport coût/performance, la contrainte de coût plus faible étant la plus justifiée compte tenu des performances des circuits modernes. Le progrès viendra surtout d'une meilleure architecture de la machine, celle des systèmes actuels de traitement étant susceptible d'évolution, d'autres apparaissant maintenant avoir atteint la limite de leur utilité. La plupart des machines actuelles ne fonctionnent en effet qu'au prix d'un monceau de software, celui-ci devant essayer de compenser les défauts de structure. Il faut espérer que les architectures futures tiendront davantage compte des besoins des programmeurs et seront basées sur une meilleure compréhension de bases théoriques du calcul. Le but futur est d'intégrer proprement le hardware, le software, et les éléments humains pour former un système sûr et utile.

L'essence du métier est la complexité et non les circuits intégrés, la mémoire, les systèmes de commandes, les compilateurs et autres spécialités, qui sont seulement des composants d'un assemblage complexe. L'accroissement de la complexité est rendu possible par l'introduction d'interfaces successives assurant les liaisons entre les différents éléments et brisant le travail total en composants assemblés hiérarchiquement, qui peuvent être manipulés par des individus ou des groupes, ceux-ci pouvant introduire périodiquement de nouveaux équipements, suivant les besoins de leur travail.

La généralisation des machines géantes est liée à des considérations économiques, l'accroissement du coût du traitement variant en fonction de la racine carrée de la puissance du système et les utilisateurs demandant un temps d'exécution plus court et des mécanismes plus puissants.

Il y a couramment plusieurs grands systèmes de traitement de l'information, parmi ceux-ci un nombre appréciable est de type temps réel et on attend d'eux de réaliser une large variété de fonctions, par exemple : le traitement du trafic d'information, le stockage et la récupération d'information, la simulation, les calculs mathématiques généraux, etc... ainsi que des combinaisons de ces éléments.

Ces systèmes de traitement de l'information doivent être capables de répondre aisément et rapidement aux changements des demandes opérationnelles. En effet, leur utilité dépend largement de la souplesse d'adaptation aux changements soudains et inattendus dans la nature de l'information aussi bien que dans la politique des utilisateurs.

Le calcul, généralement, inclura plusieurs grands calculateurs digitaux et un ensemble complexe de circuiteries périphériques comprenant des équipements de visualisation, consoles d'opérateurs et de maintenance, des équipements de commutation et communications, etc... Les caractéristiques grossières du traitement de l'information d'un tel système de calcul devront généralement permettre :

- a) - l'usage bien intégré de nombreux composants sûrs pour des raisons d'économie et de sécurité,
- b) - la capacité d'évolution constante, le système devant posséder un haut degré de potentiel de croissance de façon que le développement du système ne demande pas de "réusinage",
- c) - la capacité de faire face à un extrêmement grand volume de données. L'organisation des données sera complexe avec une grande variété de types de données. Le fichier de données est dynamique et sujet à un constant renouvellement. Le système d'Entrées/Sorties devra avoir une souplesse et une capacité sans précédent, sans interrompre le cours normal de données et les procédures de calcul, et devra avoir des réserves pour de nombreux accès temps réels sensibles à des commandes aléatoires externes au système,
- d) - des opérations en temps réel. Une réponse rapide équilibrant le coût, requiert une utilisation entière et efficace de tous les composants,

- e) - des communications automatiques prédominantes parmi les individus et les groupes et parmi les composants de la machine,
- f) - une sûreté absolue. Aucun élément simple ne peut effectuer une fonction unique de telle façon que son défaut puisse entraver le système entier, mais au contraire doit au plus diminuer la capacité du traitement,
- g) - les réponses aux réquisitions et demandes du monde réel et la capacité de se planifier dynamiquement en accord avec des règles de priorité.

Pour la réalisation des conditions générales ci-dessus, le concept actuel de l'unité de calcul simple ou centrale est frappé rapidement d'obsolescence simplement parce que les demandes dépassent ce que l'état de l'art en circuiterie a été capable de produire, particulièrement dans les temps de réponse, la sûreté et la capacité (par exemple en 1 ns, la lumière parcourt 30 cm dans le vide). Une nouvelle architecture et une nouvelle conception de la programmation sont requises pour faire face à ces desiderata. Les plus grosses unités, ayant grandement accru les possibilités de calcul résultant des techniques logiques traditionnelles, peuvent offrir un refuge temporaire mais ne résoudront pas le problème. De tels systèmes ne semblent pas devoir offrir un quelconque avantage dans le domaine de la sûreté, du coût et du potentiel de croissance (STAR, CDC 7600).

Des organisations de systèmes caractérisées par une multitude d'unités de contrôle indépendantes, sont envisagées afin de fournir la satisfaction ultime aux demandes pour une capacité et une faculté largement accrues.

Plusieurs raisons existent pour justifier un tel système. Les caractéristiques de sûreté et de possibilité d'expansion en elles-mêmes sont des raisons suffisamment valables pour pouvoir progresser au-delà du concept d'un simple calculateur. Cependant, le concept de

calculateur en duplex a été développé d'un point de vue de sécurité tandis que le système primitif d'Entrées/Sorties connecté est certainement capable d'extension.

L'aspect des applications mentionnées est une juste motivation, c'est-à-dire la détermination des classes de problèmes auxquelles nous destinons le calculateur. Il est bien connu que, en théorie du moins, il est possible pour tout calculateur de résoudre tout problème de calcul si un temps suffisant lui est alloué. En conséquence, le calculateur envisagé doit résoudre au moins quelque classe de problème dans un temps limité donné, limite qui serait dépassée par tout calculateur simple travaillant sur le même problème. Il doit aussi, au moins pour une portion du temps, être utilisé pour exécuter une tâche qui requiert tout ou partie de l'équipement disponible, sinon plusieurs calculateurs plus petits opérant de façon autonome atteindraient le même but. De plus, la fréquence d'utilisation de tout ou partie de la machine pour un problème donné doit être suffisamment élevée pour justifier une telle approche, sinon du point de vue du temps, le système primitif et coordonné d'Entrées/Sorties suffirait.

Cependant, on peut diviser les problèmes en deux classes :

- ceux nécessitant une grande quantité de transferts de données en entrée-sortie (en incluant la communication entre unités) par rapport au nombre d'opérations internes et qui bénéficieraient le plus d'un système qui minimiserait le temps de communication inter-unités.
- ceux où la quantité de transferts de données est faible en comparaison du calcul, alors la nécessité de communications de données à grande vitesse ne serait pas si impérative.

Dans le premier cas, le temps est épargné en approvisionnant par transfert rapide de grands volumes de données, mais cette provision serait inefficace dans le dernier cas, particulièrement si la dépense en temps dans la préparation des données pour le transfert est un pourcentage appréciable du temps de transfert total. Il est tout à fait probable que les travaux à exécuter seront des deux types, et, en accord avec l'organisation du système, consommeront inévitablement plus de temps pour l'exécution de l'un des types que pour l'autre. En général, tout ce qu'on peut dire est que l'organisation du système sera telle que l'efficacité de la circuiterie et le temps seront bien meilleurs que ce qui pourrait être exécuté sur un calculateur simple ou des groupes de traitement autonomes. Des mesures montrent que le rapport nombre d'instructions traitées sur nombres de bits transférés en entrée-sortie est égal à 1 et cela sans qu'aucune justification théorique n'ait pu être établie, ni que cette loi ait été contredite. En d'autres termes, le système d'Entrées/Sorties doit avoir une puissance au moins équivalente à la puissance de l'unité de calcul.

Très souvent le projet d'un calculateur ou système de traitement de données dérive d'un jeu particulier d'exigences d'un problème ou d'une classe de problèmes et la machine est étudiée pour être la plus utilisable dans leur exécution. Certaines caractéristiques peuvent être ajoutées pour rehausser les performances de la machine pour des travaux collatéraux pour un moindre coût supplémentaire. Il existe des machines à calculer prévues pour des applications générales (traitement de données) avec des extensions, par exemple au traitement de données de communication, ou pour des applications scientifiques (machine arithmétique) avec des extensions par exemple au traitement de données radar mais usuellement non des deux types (par exemple ILLIAC IV, SOLOMON). Autrement dit, il est difficile de concevoir une machine qui puisse manier des travaux largement différents avec une facilité égale, mais une exigence minimale initiale est qu'elle soit capable de manier un spectre de travaux beaucoup plus large qu'une simple machine, mais cependant pas si large que son efficacité hors tout dans l'exécution d'une de ses tâches ne soit

sérieusement réduite. On doit donc avoir la machine la plus générale possible, capable cependant d'exécuter des travaux particuliers avec efficacité.

La question de savoir jusqu'à quel point un calculateur utilise sa circuiterie de la façon la plus efficace est secondaire par rapport au facteur temps. Cependant il est possible qu'il soit inefficace à cause de l'équipement dans l'exécution de certains problèmes. De là, les différentes approches possibles pour un tel calculateur : si la classe de problèmes est restreinte et assez bien connue par avance, il est raisonnable d'intégrer un petit nombre d'unités spécialisées pour minimiser l'inefficacité de l'équipement et pour mécaniser les facilités de communication entre les unités afin d'optimiser le temps de transfert de l'information. Cela est spécialement vrai de tâches ayant une haute fréquence d'utilisation et une haute priorité. Si la souplesse est recherchée, alors il est nécessaire d'utiliser des machines universelles et l'inefficacité qui s'ensuit dans l'exécution de certains travaux doit être acceptée.

Dans l'état actuel de la technologie, la seule voie dans laquelle on peut effectivement diminuer le temps d'exécution est celle du traitement véritablement simultané des différentes phases d'une même tâche. Cela implique un système de traitement multi-unités avec un système de communication intégré pour minimiser le temps total d'exécution et de transfert. Cette division de la charge du travail, pour diminuer le coût en temps, en accord avec les caractéristiques de sûreté et de croissance, émerge comme la justification originelle d' un tel système qui permet plus de calculs par unité de temps pour un prix donné.

L'idée de diviser la charge de travail amène à penser qu'une telle division dépend en fait du problème et de la machine :

- cela dépend du problème parce que celui-ci doit être séparable de telle façon qu'une opération simultanée sur les différents segments soit possible. Les classes de problèmes de ce type favorisent un système de multi-calculateurs composé de machines universelles.
- cela dépend de la machine en vertu du fait qu'une majorité des segments de problèmes, qui résultent d'une division de ceux-ci, sont le plus vraisemblablement uniques et requièrent ainsi des caractéristiques de machine opérationnelle tout à fait distinctes qui favorisent l'utilisation de calculateurs spéciaux. Toutefois, le même point de vue a conduit aussi au projet d'une machine spécialisée comme moyen d'accroître l'efficacité. Le défaut ici est qu'il y a des segments de problèmes qui ne s'ajustent pas aisément aux possibilités de n'importe quelle machine spécialisée, ce qui confirme la conclusion précédente.

Une alternative est de diviser la structure pour optimiser certaines opérations. Par exemple une machine sera utilisée pour l'édition, conversion de données, tandis qu'une autre ferait les comparaisons et une autre les opérations arithmétiques et logiques (ex. GAMMA 60). Cependant, toutes ces fonctions font partie des capacités d'un calculateur universel dont, à cause de sa souplesse, on peut attendre de manier une plus grande variété de fonctions avec une efficacité accrue.

CHAPITRE II

INTERCONNEXION ET CADENCEMENT DU SYSTEME

Nous avons présenté dans le chapitre précédent certains avantages et inconvénients de l'emploi des ordinateurs spécialisés et généraux en tant que constituants d'un complexe multi-ordinateur, les avantages d'un système composé de machines universelles l'emportant sur les inconvénients. Quelques considérations générales ayant rapport à la mécanisation de l'organisation d'un multi-calculateur basé sur l'utilisation de machines universelles sont proposées dans cette section et nous verrons par la suite comment nous avons cherché à répondre à l'organisation du système décrite ici, qui est hypothétique mais sert à illustrer comment la tentative de conception d'un système lié à ce problème peut donner des résultats intéressants.

Il est fondamental pour un système multi-calculateur, qu'il y ait une coopération complète entre les calculateurs sous le contrôle d'un superviseur qui inscrit les travaux et donne les affectations. Le calculateur individuel est alors déchargé de la nécessité de s'approvisionner en programmes. Cela suggère deux choses :

1°/ - Une combinaison de mémoires modulaires de telle façon que la mémoire du système soit un agrégat de plusieurs unités mémoires plus petites par opposition à une grande mémoire commune à tous les calculateurs. Une telle combinaison permet le transfert parallèle de grands volumes de données par transfert électronique d'un ou plusieurs modules de mémoires d'un calculateur à un autre. Cette forme de transfert de données s'oppose à la pratique courante de transferts en série d'une machine à une autre, soit directement, soit par le canal

d'Entrées/Sorties, ou des nombreux calculateurs qui effectuent ce transfert par le moyen d'une mémoire rapide intermédiaire. Il est de plus souhaitable d'avoir des blocs mémoires disponibles à tout moment, de telle façon que le nombre d'unités mémoires modulaires soit plus grand que le nombre de calculateurs. Le principal avantage d'une telle organisation est que les programmes et les données peuvent être partagés entre plusieurs calculateurs.

2°/ - Des équipements périphériques totalement partagés mais de telle façon que n'importe quel moyen périphérique soit disponible pour un seul calculateur à un instant donné sur une base de priorité. L'avantage manifeste d'un équipement périphérique partagé est la disponibilité instantanée de vastes fichiers de données pour tous les calculateurs sans avoir à les reproduire.

La réalisation d'un système de calculateurs ayant les caractéristiques génériques désirées, dépend totalement du développement d'un système satisfaisant de communication inter-modules ou de commutations de données qui soit sûr, aisément élargissable et économique. Une fois qu'un tel modèle d'interconnexion est adopté, quelques aspects de programmation et d'opération logique globale doivent être spécifiés en termes du système d'interconnexion.

2.1 - SYSTEMES D'INTERCONNEXION

Le système d'interconnexion le plus simple est conceptuellement un réseau de commutation spatial comme il est indiqué sur la figure 1 dans laquelle les modules de calculs sont connectés aux modules mémoires par contrôle des croisements d'une matrice de commutation. La simplicité conceptuelle, cependant, est compensée par l'accroissement de la complexité opérationnelle et de construction. Un circuit de portes contrôlées est requis à chaque croisement de la matrice (ou à chaque niveau) par digit et par fonction et une relation de produit existe entre le nombre de calculateurs, de mémoires,

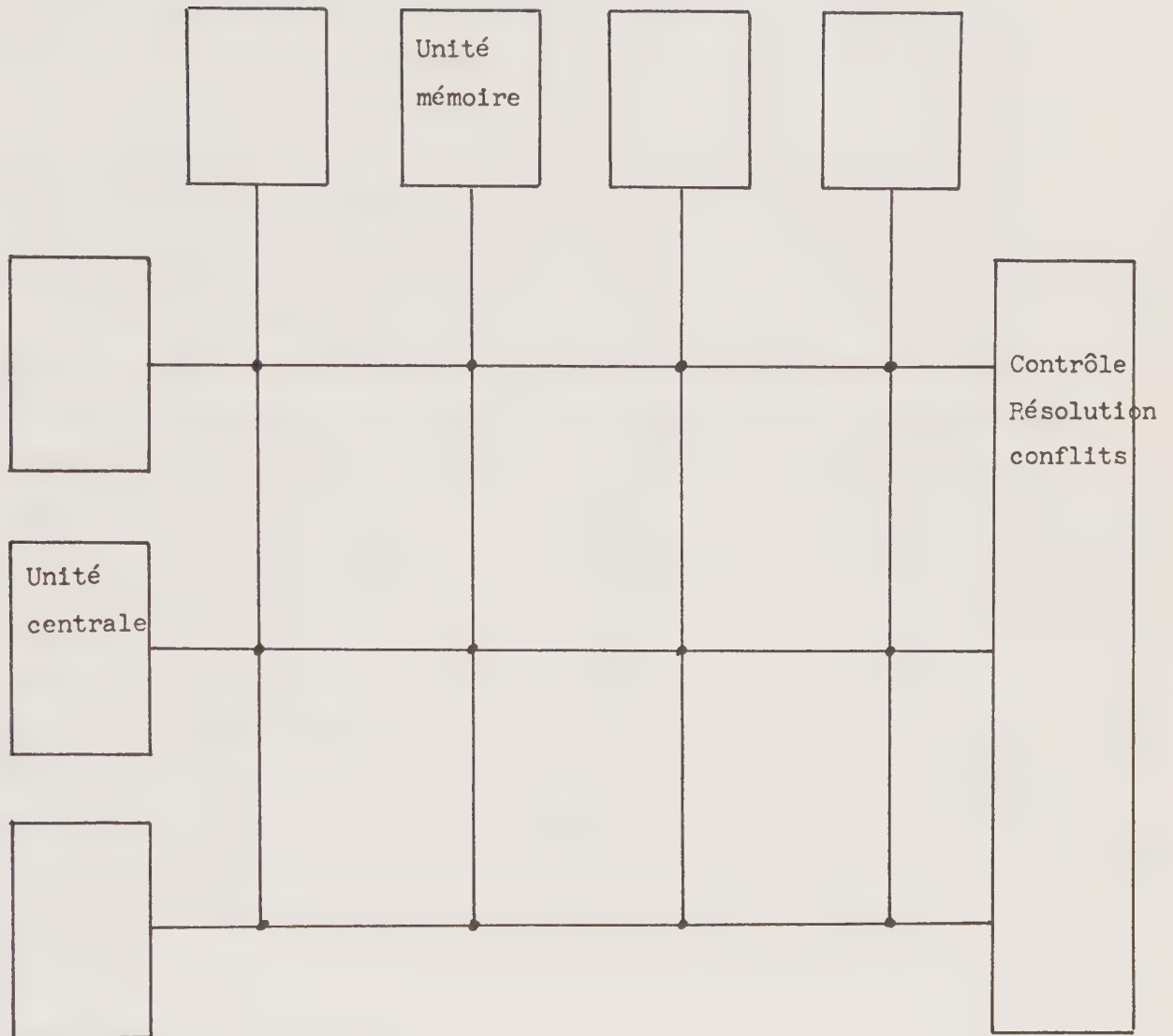


Fig 2-1

d'équipements périphériques et de digits d'information. Le temps de réponse excessif des éléments physiquement réalisables des portes de croisement limite sévèrement la vitesse maximale à laquelle les modules peuvent être commutés. D'ailleurs, du temps supplémentaire doit être dépensé pour s'assurer que la connexion peut être effectuée et réaliser d'autres fonctions de la logique de contrôle des commutations. Une fois une connexion de croisement établie entre deux modules du système, elle doit être maintenue pendant une période étendue afin que l'investissement en temps soit rentable. Le rendement du système dépend aussi des dimensions du réseau dans une proportion aussi importante que la vitesse mémoire est sensible au nombre de mots contenus dans un bloc. En conséquence, le cadencement du système demande un réajustement à chaque croissance de ce système. Pour surmonter ces problèmes, on peut évidemment calculer le réseau pour le système le plus étendu, ce qui veut dire qu'il ne peut facilement être modularisé. Il peut alors être nécessaire d'avoir de la circuiterie ou de la programmation supplémentaire pour résoudre les possibilités de conflits de la matrice et pour manipuler les priorités, en plus de la nécessité d'ordonnancer le réseau.

En général, il est impossible pour un programmeur de s'être arrangé pour que toutes les données et les instructions (ou une grande majorité de celles-ci) et ceci pour un calculateur donné, résident dans un seul module mémoire. En conséquence, les possibilités d'accès à une multi-mémoire, pour un tel système, doivent être réalisées avec une grande vitesse de base. Un système de commutation croisée ne possède pas cette caractéristique de vitesse.

Pour vaincre cette difficulté, l'accès du calculateur vers plusieurs mémoires peut être accompli en gardant les connexions-calculateur déjà établies à la demande. Pour empêcher toutes les mémoires de répondre à toutes les demandes d'adresse, l'information du calculateur doit être codée de telle façon que seule la mémoire particulière spécifiée par le code réponde. Mais si la mémoire est codée, il n'est pas nécessaire d'établir des connexions croisées

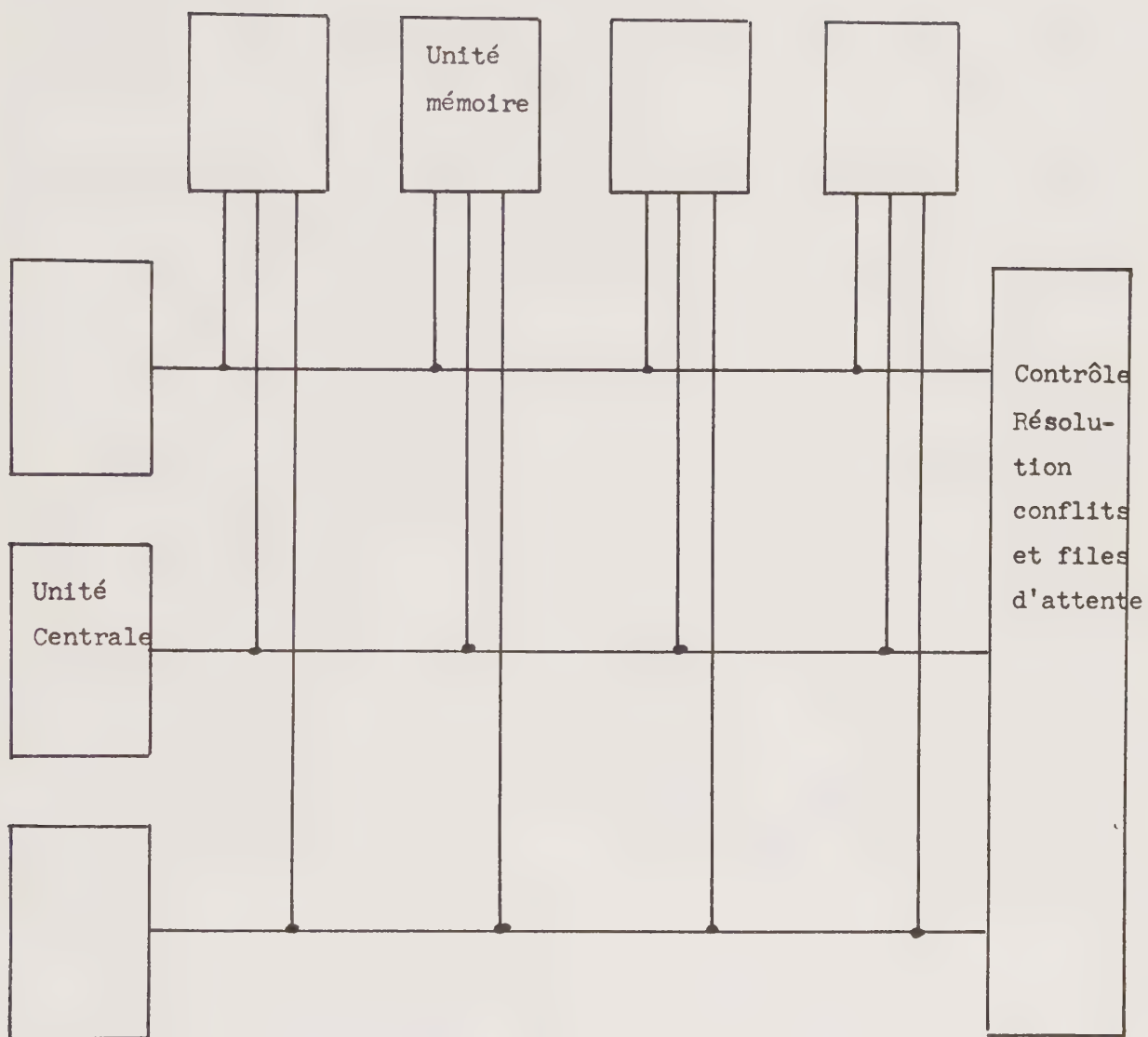


Fig 2-2

calculateur à mémoire, c'est-à-dire qu'une connexion permanente peut aussi bien avoir été faite ; en conséquence, la configuration du système peut être réduite à celle indiquée sur la figure 2. Cet arrangement demande encore un quelconque type de système de file d'attente pour empêcher l'accès mémoire simultané par plus d'un calculateur en plus d'une combinaison pour la détermination des priorités. Sur la figure 2, on a montré plusieurs calculateurs ayant des accès indépendants à chacune des différentes mémoires, par leur propre ligne de communication. Ceci n'est pas souhaitable parce que chaque mémoire demande une circuiterie d'adressage séparée pour chaque calculateur. Le coût d'un tel arrangement est prohibitif et la croissance du système demande des additions de circuiterie pour chaque mémoire. Pour éliminer ce problème, la solution la plus claire est de partager la circuiterie d'adressage mémoire entre tous les calculateurs par des moyens quelconques. Si cela est fait, les lignes de communication à la mémoire de chaque calculateur peuvent aussi bien être partagées. Le système peut alors se réduire à la configuration indiquée sur la figure 3. Dans ce cas, à la fois, les mémoires et les calculateurs doivent être codés pour assurer le courant correct des données et les combinaisons de priorité et files d'attente doivent aussi être employées de façon telle qu'un seul et unique calculateur puisse avoir accès à la ligne à un instant donné et s'assurer que seulement la ou les mémoires spécifiées sont atteintes par l'information provenant du calculateur.

Un problème sérieux peut s'élever avec cette configuration à cause de la possibilité d'un calculateur d'immobiliser complètement le moyen de communication aussi longtemps qu'il est nécessaire pour exécuter son travail. Une telle possibilité de bloquer les autres calculateurs limite sérieusement la souplesse du système. Cette difficulté peut être dominée en imposant que chaque calculateur soit autorisé à accéder à la voie de communication d'une manière séquentielle, ce qui est possible parce que le temps de fonctionnement logique d'un calculateur est, de façon significative, inférieur à la durée

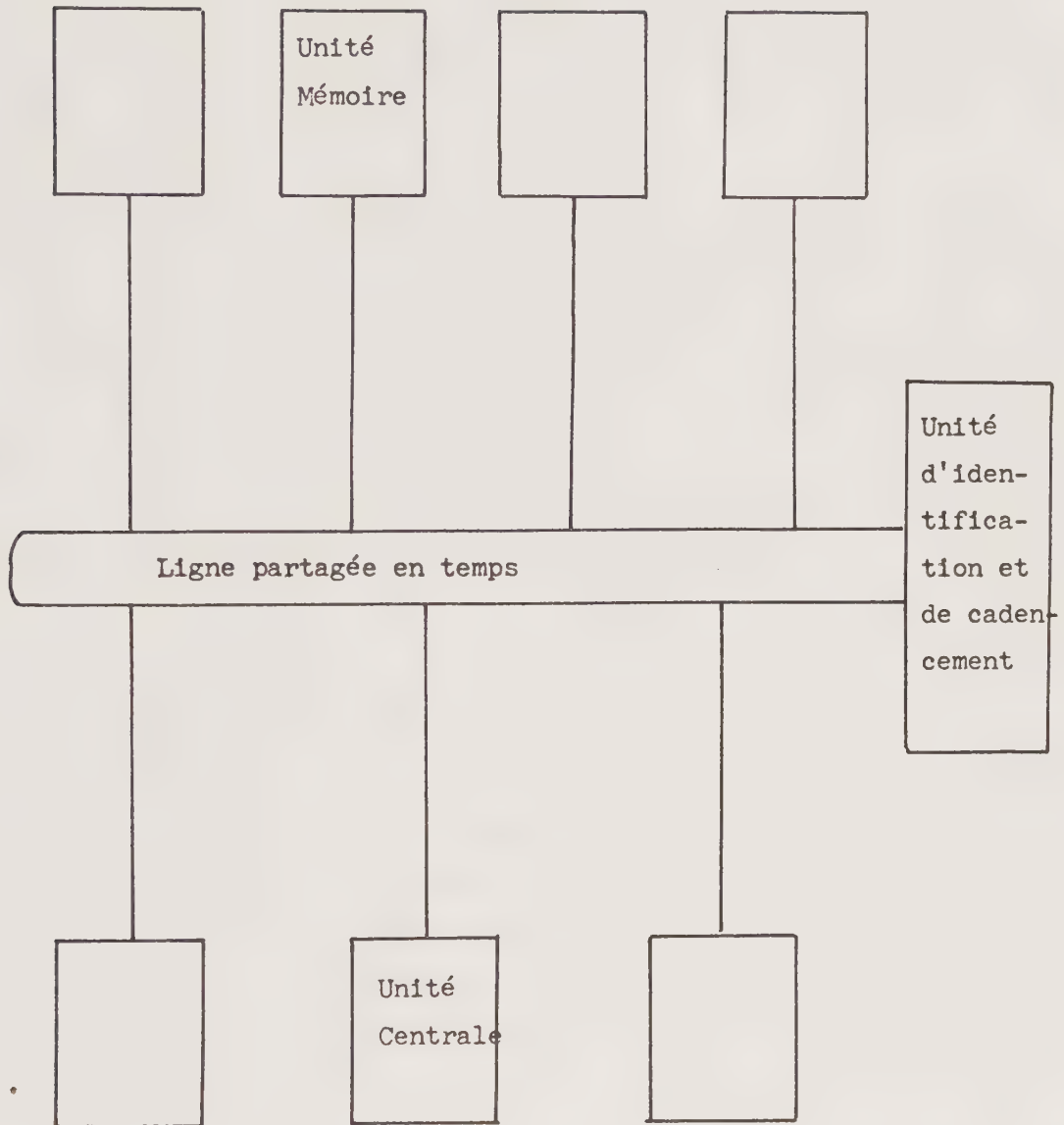


Fig 2-3

du cycle mémoire (d'un facteur 15 à 30), de telle façon que le calculateur peut adresser sa mémoire, puis abandonner la ligne omnibus pour utilisation par les autres, chacun à son tour, tandis qu'il attend la génération de la réponse de la mémoire. Un cycle mémoire plus tard, la réponse est obtenue, au moment où le calculateur a de nouveau accès à la ligne (un exemple est donné par le "tonneau" du CDC 6600). Puisque chaque calculateur doit avoir accès à sa mémoire (par la ligne omnibus) au moins une fois par cycle mémoire, un arrangement convenable est de permettre l'accès à la ligne à chaque calculateur au moins une fois par cycle dans un intervalle de temps (T_M/n) égal au temps de cycle mémoire (T_M) divisé par le nombre de calculateurs (n) demandant accès à la ligne. Ce concept de division en temps du moyen de communication est une extension de la technique de temps partagé bien connue qui a été appliquée aux lignes de transferts pour permettre à la mémoire rapide d'être utilisée comme tampon d'entrée-sortie. L'identification du calculateur sera établie par l'instant d'arrivée sur la ligne. Les conflits sont réglés automatiquement par cette technique.

En bref, il y a trois moyens de base pour interconnecter les modules, à savoir : division de l'espace (matrice), division en temps, division en fréquence. Les avantages de la méthode matricielle et les différents degrés de raffinement conduisant au concept de la division en temps ont été discutés. Les plus grandes motivations existent donc pour rejeter un arrangement spatial, le temps de transfert excessif et la complexité de la circuiterie étant inclus par opposition à une économie attendue en temps et équipement quand les lignes de communications sont partagées en temps. L'ordre des économies à attendre de l'emploi de la division du temps sur la division de l'espace est aisément calculable pour un état donné. Cependant, une solution mixte utilisant les deux concepts espace et temps peut être employée, c'est la solution du multiplexage.

Si l'on considère une division fréquentielle qui offre une grande souplesse, il est alors nécessaire d'avoir une fréquence dans la gamme de 200 à 500 MHz. La circuiterie de modulation et démodulation dans cette gamme, pour un grand nombre de digits d'information, et la nécessité d'un changement rapide de la fréquence centrale des filtres à large bande, sont une limitation de l'emploi de cette technique dans le domaine des calculateurs.

Pour le système, le nombre de calculateurs peut varier d'au moins deux (pour pouvoir être appelé un multi-calculateur) à un nombre maximum quelconque prédéterminé, limité par des considérations de circuiterie physiques qui déterminent le temps minimum pendant lequel il est possible de transférer de l'information avec succès d'un point du système à un autre. Le temps dépend du temps de montée pouvant être obtenu des portes logiques, du temps de positionnement d'un registre et du temps de réponse d'une ligne de transmission.

Le mode de division du temps postulé jusqu'ici est séquentiel et basé sur la segmentation du temps de référence de l'unité (temps de cycle mémoire). Ce temps de cycle mémoire peut être en rapport avec le temps minimum nécessaire pour effectuer une opération de décision qui correspond à l'enchaînement de deux ordres successifs (cycle mineur). Ce temps a été convenablement choisi dans le dessein de l'identification d'un calculateur par simple coïncidence de temps. Cette forme de division du temps est souhaitable pour le transfert de données calculateur à mémoires à cause de la nature précise des mémoires et des calculateurs. Cependant, une division en temps non séquentielle peut être utilisée comme moyen de communication avec des équipements, tels les disques et bandes. En effet, une telle méthode est plus souhaitable dans le cas des équipements d'entrée-sortie.

Dans le système envisagé, nous utiliserons un système de multiplexage, les unités s'interconnectant par l'intermédiaire de canaux ou de registres de liaisons. Les canaux sont du type pipeline. Leur

nombre est fonction du nombre d'éléments présents. Certains de ces pipelines sont cependant spécialisés pour les signaux de commande ou pour les transferts de données. Le cadencement se fait par l'intermédiaire d'horloges déclenchées par certains signaux.

2.2 - DISTRIBUTION DU SIGNAL

Il est important d'observer que la communication calculateur à mémoire doit arriver dans un intervalle de temps qui est une quelconque fraction intégrale du cycle mémoire, par exemple T_M/n , où T_M est le temps de cycle mémoire et n le nombre de calculateurs. Le véritable problème avec la matrice d'espace qui a conduit au concept de division du temps, est le temps excessif nécessaire pour établir le lien de communication entre un calculateur et une mémoire, mais c'est parce que la connexion a été établie par le moyen d'une structure fixe à portes qui est de façon inhérente limitée en vitesse et surtout parce que l'on doit résoudre, par des logiques spéciales ou des programmes consommant du temps, les conflits de la matrice, les priorités de connexion, les signaux d'activité et la sélection des chemins alternés. La division en temps résoud d'elle-même tous ces problèmes exception faite de la limitation physique due à la distribution du signal.

Par conséquent, la réalisation d'un système de distribution omnibus sur lequel peut être transmis sans détérioration appréciable une information à un taux de pulsation rapide, est essentiel. Les caractéristiques du signal transmis devront être indépendantes du nombre d'équipements destinés à recevoir l'information. Comme il a été dit plus haut en rapport avec la méthode matricielle, cette simple exigence écarte l'utilisation d'une structure de communication par portes. Le moyen le mieux connu pour réaliser le degré de performance désiré est la ligne de transmission linéaire adaptée.

Dans un but d'extrême sûreté, le système de transmission en entrées doit être en duplex, autrement certaines parties nécessitent d'être doublées . Cela est aussi demandé au modèle matriciel, mais c'est, de façon significative, moins cher pour le système à division en temps. Le modèle omnibus employant moins de composants, la sûreté est également plus grande.

Au regard de la possibilité d'expansion, un tel système est idéal. Puisqu'un système de lignes de transmission sera adapté (à travers une circuiterie de transmission de ligne appropriée, qui connecte les circuits à portes du système vers la ligne accordée), une charge constante est présentée aux circuits de transmission sans tenir compte du nombre d'équipements à la fin de la ligne. Dans ce cas, le cadencement du système et le projet de circuiterie transmetteur sont fixés pour toutes les configurations possibles du système. Des équipements peuvent être ajoutés à une installation minimale en fournissant les sections nécessaires de lignes de transmission. La variation des temps de retard quand le système croît est sans importance dans la mesure où les délais associés au système maximum sont tolérables.

2.3 - CADENCEMENT DU SYSTEME

Un aspect important est le cadencement du système.

Le cadencement caractéristique sera produit par un générateur de temps principal qui fournit des pulsations phasées aux unités de calcul. Ces pulsations permettent aux différents calculateurs de s'accorder sur l'horloge principale afin de permettre la reconnaissance des demandes par la position dans le temps de l'impulsion. Les pulsations pourraient être générées dans un code Gray pour lever les ambiguïtés ou les troubles transitoires quand la pulsation change. Ce code peut être à n digits (on peut alors identifier 2^n calculateurs). La reconnaissance du code permet à la mémoire de déterminer l'origine

de la demande et de placer en conséquence son information à l'instant approprié. Il sera important de limiter les décalages dus aux retards sur la ligne afin de préserver l'intégrité de l'information. Les impulsions de référence seront retardées soit au calculateur, soit aux mémoires, de préférence au niveau des mémoires puisque les différences dans les caractéristiques de cadencement des mémoires, dues aux tolérances, doivent être prises en compte de toute manière.

2.3.1 - Cycle machine

Le cycle machine a une très grande influence sur les performances du système : trop court, il risque de poser des problèmes de longueur de câble et de temps de retard, donc de contrôle. Trop long, il entraîne une segmentation trop importante de logique puisqu'il faut alors, pour conserver les mêmes performances, découper davantage le système.

Ce cycle doit, d'autre part, être calculé pour unifier les temps de travail de la mémoire principale et de l'unité de commande.

Compte tenu du niveau atteint par la technologie, le cycle machine peut varier entre 25 ns et 100 ns, temps imparti au décodage de signaux (ouverture d'une dizaine de portes).

Nous choisissons un temps de cycle de 50 ns, ce qui implique une logique à 4 ns, ce qui est réalisable actuellement sans trop de problème. Le choix de la technologie (intégrée ou discrète) est important pour les problèmes d'échange thermique; alors que certains prônent les circuits LSI, d'autres pensent que les circuits discrets sont plus sûrs à l'heure actuelle. Nous n'en discuterons pas plus avant, notre travail restant sur un plan théorique.

2.3.2 - Cycle lecture/écriture

Le choix de la technologie, pour la mémoire centrale, impose un cycle d'écriture/lecture. Si ce cycle est trop faible, le gain attendu de la réduction du temps peut être insuffisant vis à vis du coût de cette mémoire. Autrement dit, la rapidité de la mémoire principale peut varier dans des proportions notables impliquant simplement une segmentation plus ou moins importante de l'accès. Dans ce cas, simplement, la méthode d'accès risque d'être très lourde.

La vitesse actuelle des mémoires à tores de ferrite permet des temps d'accès de 500 ns et un temps de lecture/écriture de 750 ns. Nous choisissons ces deux grandeurs comme caractéristiques de la mémoire principale.

Ceci entraîne l'existence de $(750/50)$ unités distinctes possibles, soit 15 au maximum.

CHAPITRE III

DESCRIPTION DU SYSTEME

Le calculateur envisagé est un calculateur de type universel pouvant faire face à tous les types de traitement par simple adaptation pour répondre aux critères définis plus haut.

Le principal intérêt réside dans les problèmes de transferts de données et de contrôle d'un point à l'autre du système.

Il est important de noter que la circuiterie doit être prévue pour faciliter la programmation et éviter les blocages. Cette nécessité de tenir compte de la dualité circuiterie-programmation implique un certain nombre de solutions :

- La circuiterie ne doit pas être telle que la gestion du système consomme beaucoup de temps.
- Inversement, la facilité de programmation ne doit pas conduire à une structure rigide et coûteuse de la circuiterie.

La modularité est une des solutions qui peuvent être envisagées pour résoudre ce problème. En effet, chaque fonction peut alors être décomposée sous forme de boîtes noires réalisant une "boîte à outils" dont les éléments serviront suivant les besoins.

3.1 - CARACTERISTIQUES OPERATIONNELLES

Les unités centrales des calculateurs incluent des équipements à la fois pour le cadencement des instructions et pour la manipulation des données. Par conséquent, chaque calculateur doit contenir tout l'équipement traditionnel pour l'exécution d'une instruction. Il doit être possible de programmer les différents calculateurs pour réaliser une large variété de fonctions qui évitent une division interne du travail parmi les calculateurs identiques, de nature logique ou arithmétique.

Les calculateurs travaillant indépendamment peuvent communiquer par transfert mémoire de telle façon que chaque calculateur agisse sous le contrôle du programme et qu'il y ait transfert mémoire à la demande pour empêcher des interférences catastrophiques entre calculateurs. Chaque calculateur à un instant donné n'a accès qu'aux modules mémoires qui lui sont liés par affectation. Il peut accéder dans le temps qui lui est imparti sur la ligne omnibus aux autres modules s'il y est autorisé.

Les mémoires multiples affectées au calculateur peuvent être distinguées les unes des autres par le numéro d'affectation contenu dans un registre d'identification. Une conséquence significative est que plusieurs mémoires peuvent recevoir le même numéro d'affectation, elles forment alors une pile de copies identiques. Les données et les instructions peuvent donc être dupliquées par le système sans accroissement du temps.

Le nombre de digits nécessaires pour adresser un mot quelconque en mémoire n'est pas plus long que le nombre nécessaire pour l'adressage dans un calculateur simple ayant la même capacité mémoire. Seul le découpage interne varie. Sous contrôle du superviseur, une unité mémoire peut avoir son association, avec un calculateur donné, changée avec un autre calculateur en modifiant son code d'affectation,

c'est-à-dire le contenu du registre de liaison, mais cela ne peut être accompli que par le superviseur de telle façon qu'aucun calculateur ne puisse se transférer arbitrairement sans la coopération du superviseur.

Il faut souligner que les modules sont strictement identiques de telle façon qu'ils soient complètement interchangeables. Quoique à un instant donné, un total limité d'instructions ou de données puisse être obtenu (à cause de la possibilité limitée d'adressage), des transferts mémoires simples entre calculateurs fournissent une capacité presque illimitée. Tout calculateur a une accessibilité vers tous les modules mémoire qui est de beaucoup supérieure à sa capacité inhérente d'adressage.

Une position mémoire peut être changée (par le superviseur) en modifiant le contenu du registre indicateur et peut donc être affectée à un autre calculateur. Avant d'envoyer une mémoire vers un calculateur, le superviseur doit insérer les mots clés dans les registres adéquats. Toute demande de mémoire supplémentaire devra passer par le superviseur qui affectera les blocs suivant les disponibilités. La taille des blocs mémoire dépend de plusieurs facteurs : segmentation logique de la mémoire, niveau d'adressage, mémoire minimum requise pour effectuer un problème, etc... Un compromis doit être trouvé parmi les différentes possibilités.

Puisque les mémoires ne sont pas affectées de façon permanente à un calculateur et n'occupent pas de positions fixes dans la structure, elles doivent être identifiées par un moyen quelconque qui sera en principe l'affichage par un bouton d'un numéro physique pouvant être testé par le superviseur et par le calculateur de contrôle.

3.2 - COMPOSANTS DU SYSTEME CENTRAL

Le système central sera composé de plusieurs modules indépendants en liaison par différents canaux ou registres frontière. La nécessité de maintenir le système en état malgré la défaillance d'un élément implique une multiplication des éléments.

3.2.1 - Unité de commande

L'unité de commande joue le rôle de répartiteur de l'information dévolue en partie à une unité centrale. Son but est d'ordonnancer les demandes d'un programme et de diriger les informations. Le système comportera de 2 à 8 unités de commande notées UC0 à UC7. Le numéro logique sert de repère au système. Ce nombre (8) n'est pas arbitraire. Nous avons dit que le nombre de calculateurs était inférieur ou égal à 15 en fonction du rapport cycle écriture/lecture sur cycle mineur. Cependant, si nous nous bornons seulement au temps d'accès, soit 500 ns, le nombre de calculateurs tombe à 10 au maximum. Nous choisissons la puissance de 2 la plus proche, soit 8 comme nombre de calculateurs, ce qui donne :

- 400 ns de temps de rotation du système,
- la possibilité de faire la demande dans un temps moitié du temps de lecture/écriture (750 ns).

D'autre part, chaque unité de commande travaille en multiprogrammation sur un certain nombre de programmes. Ce nombre de programmes a été fixé à 8 parce que la commutation et l'exploration du système demandent 100 ns, soit 800 ns pour un cycle complet.

Le nombre total de programmes sous contrôle en cours d'exécution est donc de 64. Il ne faut pas oublier que la puissance du système d'entrée-sortie doit être du même ordre que celle du système central, ce qui donne un total de 64 contrôles pour le système d'entrée-sortie.

3.2.2 - Mémoire locale

Le système utilise une mémoire locale rapide comme tampon entre l'unité de commande et l'unité mémoire afin de fournir les informations à un rythme suffisant pour alimenter le système. Cette mémoire a un cycle de lecture/écriture de 100 ns. Nous expliquerons plus loin son rôle et son mode de fonctionnement.

3.2.3 - Unité microprogrammée et microprogrammation

A chaque unité centrale est associée une unité microprogrammée, qui renferme les microprogrammes de la machine. Une partie seulement des instructions est sous forme de microprogrammes, les autres instructions étant câblées. La micro-mémoire associée a un cycle de lecture de 100 ns.

Cette micro-mémoire n'est pas modifiable. Cependant, il est possible d'écrire des microprogrammes au moyen d'un mécanisme spécial lié à l'existence de la mémoire locale et sur lequel nous reviendrons plus loin.

3.3 - UNITE D'EXECUTION

Chaque fonction principale (logique, arithmétique fixe, arithmétique flottante, etc...) est indépendante et enfermée dans un bloc spécifique. Les différentes unités d'exécution sont communes aux unités centrales. Chaque fonction est en nombre variable afin d'adapter rapidement le système aux besoins nouveaux. Un distributeur attribue le travail au premier bloc disponible. La panne d'une unité réduit partiellement la puissance du système sans l'arrêter complètement.

3.4 - MEMOIRE GENERALE

Chaque unité doit pouvoir accéder à un banc mémoire composé d'unités indépendantes. L'accès à ces bancs se fait par l'intermédiaire de registres de liaisons. Nous discuterons plus loin de la taille et du nombre des bancs mémoires, ainsi que du mécanisme d'accès.

3.5 - MEMOIRE AUXILIAIRE

Il est possible d'adjoindre au système une mémoire étendue de façon à augmenter sa capacité afin de diminuer les temps d'accès lors du traitement de gros fichiers. La méthode d'accès sera discutée plus loin.

3.6 - LE SYSTEME D'ENTREE-SORTIE

L'objectif de tout système d'entrée-sortie efficace est de transférer économiquement, sûrement et de façon ordonnée la plus grande quantité de données dans le temps le plus court possible avec un minimum d'interférences avec les autres opérations du système. Alors, à côté des équipements d'Entrées/Sorties eux-mêmes, la philosophie du système vis à vis du contrôle et de la communication est de première importance.

Pour des raisons d'économie de circuiterie, les transferts de données entre les équipements périphériques et la mémoire prennent place sur des lignes multiplexées en temps. Le problème réel à cet égard n'est pas l'aspect multiplexage mais plutôt son contrôle. Le contrôle et le transfert des données seront exercés par un calculateur utilisant ses propres lignes de communication mémoire. Le problème des files d'attente est minimisé en employant des lignes de transfert séparées vers les mémoires et les équipements d'Entrées/Sorties.

A cause des lignes de transmission en parallèle des données, l'objectif principal doit être d'accomplir l'utilisation la plus efficace de la ligne de transfert multiplexée en temps. Autrement dit, la ligne devra être occupée à 100 % en fournissant suffisamment de données pour le transfert puisque le problème se réduit à la capacité physique de la ligne. Il semble donc raisonnable de penser que l'usage de la ligne sera indépendant des programmes et programmeurs. Le système de contrôle optimisera les conditions de trafic.

3.6.1 - Calculateur d'entrée-sortie

Puisque le flux de données s'effectue entre les mémoires et les organes d'entrée-sortie, il est clair qu'un contrôleur est nécessaire pour effectuer les fonctions générales suivantes : reconnaître la source des demandes, manipuler les interruptions d'entrée-sortie, exécuter les instructions d'entrée-sortie, etc... Les unités mémoire ne peuvent remplir ce rôle puisqu'elles sont passives. Les unités centrales pourraient le remplir mais cela les obligerait à une gestion et aux risques d'attente liés aux événements extérieurs. Une nouvelle unité du système appelé calculateur d'entrée-sortie est introduite.

L'unité de contrôle d'entrée-sortie est un petit calculateur qui va manipuler les ordres d'entrée-sortie, les contrôles et effectuer des transferts mémoires à périphériques demandés par les programmes en unités centrales.

Puisque les entrées-sorties doivent se faire indépendamment des unités centrales, il est nécessaire d'accéder aux blocs mémoire des unités centrales par des voies séparées. D'autre part, à cause des temps de réponse des équipements d'entrées-sorties par rapport aux vitesses mémoire, il est nécessaire de prévoir un tampon. L'unité de contrôle aura un code propre et un registre contenant le numéro logique du calculateur avec lequel elle est en liaison.

3.6.2 - Organisation du système d'entrée-sortie

La raison la plus significative d'une philosophie différente pour les communications d'entrée-sortie est que le flux de données est dirigé vers une multiplicité de mémoires et une multiplicité d'équipements périphériques telle qu'on est obligé d'effectuer plusieurs opérations en parallèle. Puisque les organes périphériques ont des caractéristiques largement variables, les différents problèmes de contrôle et d'identification croissent lorsqu'ils ne sont pas aisément manipulés par des séquences d'ordres adaptés. De même que pour les transferts entre calculateurs, il sera intéressant d'utiliser le multiplexage en temps pour permettre l'utilisation maximum des canaux d'entrée-sortie.

Une séquence typique de demande d'entrée-sortie se fera de la sorte : le calculateur, à l'origine de la demande, enverra une interruption vers le système de contrôle d'entrées-sorties. Celui-ci enregistrera la demande et initialisera la séquence d'opérations. Les données en entrée ou en sortie seront extraites respectivement des équipements périphériques ou de la mémoire, puis transférées soit vers la mémoire, soit vers les équipements. En fin d'opération, l'unité de contrôle avertira l'unité centrale de la bonne marche de l'opération ou des erreurs irrécupérables. Les conflits pouvant exister entre les demandes des différents calculateurs seront réglés par une priorité affectée par le superviseur aux programmes ou fixée par l'instant de la demande.

Si le calculateur qui est à l'origine de la demande désire adresser la mémoire engagée dans un autre processus, les problèmes de priorité surgissent aussi. Ils seront alors manipulés comme précédemment en tenant compte : de l'ordre des demandes, des priorités des programmes, de la disponibilité des équipements.

Une différence fondamentale intervient aussi vis à vis des calculateurs classiques. Puisque les mémoires sont affectées à des calculateurs ou transférées d'un calculateur à un autre, si une demande prioritaire d'affectation en provenance d'un calculateur oblige le superviseur à lui transférer un bloc mémoire déjà engagé dans un processus d'entrée-sortie, il faudra que le système tienne compte de la nouvelle affectation et surtout essaye de terminer le processus afin de pouvoir transférer le contenu de la mémoire sur un équipement autre de façon à pouvoir reprendre par la suite la séquence des opérations. Autrement dit, il sera nécessaire de prévoir une zone tampon pour permettre l'achèvement des processus lorsque le demandeur a été retiré de l'état actif.

Les commandes vers les unités de contrôle d'entrée-sortie ou vers les calculateurs passent par une ligne omnibus multiplexée dans le temps, l'origine de la demande étant reconnue par la position de l'impulsion sur la ligne. Le transfert de l'information se fera comme indiqué plus haut par des lignes particulières afin de ne pas interférer avec les communications mémoires à calculateurs. Cela nécessite un tampon au niveau de la mémoire centrale pour enregistrer les différentes demandes et données. Cependant, en utilisant une partition des blocs mémoire, on peut diminuer de façon significative les possibilités de conflits.

Le choix du calculateur devant assurer l'entrée-sortie peut dépendre soit du superviseur, soit d'un mécanisme quelconque de priorités ou au contraire être affecté par association de la même manière que les unités mémoires, ce qui semble la meilleure solution.

3.6.3 - Interruptions d'entrée-sortie

Toutes les interruptions d'entrées-sorties sont maniées primitivement par les unités périphériques sur une base de priorité mais le rétablissement des demandes interrompues doit être accompli par le calculateur à l'origine de la demande. Cette méthode permet

un étagement sans limite des interruptions d'entrées-sorties sans charge supplémentaire pour l'unité de contrôle périphérique.

Les commandes d'interruptions peuvent être générées ou bien de façon interne, ou bien de façon externe par horloge ou signal aléatoire. Dans le dernier cas, un sous-programme de contrôle doit être disponible pour diriger et satisfaire la demande. L'unité de contrôle doit connaître l'origine et la classification de l'interruption. Le problème est donc de retrouver l'auteur de la demande. Pour cela, on utilise une ligne commune entre les différentes unités de contrôle sur laquelle est placé le niveau de l'interruption et son numéro. Cependant, certaines interruptions sont très prioritaires : elles seront alors directement affectées à une unité périphérique sans passer par le contrôleur d'interruptions. Le problème est donc celui de la hiérarchisation des signaux d'interruption et leur aiguillage convenable.

3.6.4 - Capacités additionnelles au système d'entrée-sortie

Dans le cas de transferts de gros fichiers, il peut être intéressant d'utiliser une mémoire commune de très grande capacité mais plus lente que celles des unités centrales, afin de permettre la constitution de vastes fichiers immédiatement accessibles.

3.6.5 - Composants du système périphérique

La gestion des entrées et sorties du système et les liaisons avec le monde extérieur doivent être indépendantes des unités de commande sous peine de voir le système inoccupé pendant ces temps d'attente. Cette gestion sera confiée à des unités périphériques en nombres variables de 2 à 16 (UPØ à UPI5) qui assurent la bonne marche des opérations et approvisionnent le système en données et transmettent les résultats sur les lignes de transfert par l'intermédiaire d'interfaces adéquates.

De la même manière et pour les mêmes raisons que plus haut, le nombre indiqué ici n'est pas complètement arbitraire. le système périphérique doit être au moins aussi puissant que le système central. Comme de plus le système périphérique assure le rôle de superviseur, il est alors évident que le nombre d'opérations simultanées de ce système doit être supérieur à celui du système central. Nous avons choisi le nombre 16 en pensant à une partition 8/8 : 8 engagés dans un processus d'entrée-sortie, 8 engagés dans les contrôles et la supervision du système.

Ces unités périphériques sont toutes identiques, ce sont des calculateurs spécialisés dans les travaux de gestion et la conduite des organes d'entrées-sorties. Ces unités périphériques ont des caractéristiques leur permettant de jouer un rôle de superviseur vis à vis des unités centrales. Elles accèdent à toutes les lignes de liaison et de commande.

Les unités périphériques sont liées à un système appelé unité de liaison canal qui assure la commande et le contrôle des canaux d'entrées-sorties et réalise les adaptations nécessaires. Toutes les lignes externes passent par cette unité de liaison canal qui est modulaire de façon à ne pas stopper le système en cas de panne partielle. Cette unité de liaison canal gère un ensemble de 64 canaux multiplexés en 64 sous-canaux, de 64 canaux à grand débit. De plus, il existe 1 024 voies d'interruptions temps réel hiérarchisée groupées 16 par 16, soit 64 groupes d'interruptions.

3.7 - UNITE DE CONTROLE CENTRALE, UNITE DE CONTROLE PERIPHERIQUE

Afin de maintenir à un haut niveau la capacité du système, il est nécessaire de détecter et corriger les pannes le plus rapidement possible. De plus, dans les temps morts, il doit être possible de tester les équipements non occupés. Ceci est réalisé par deux calculateurs ayant accès à toutes les unités à la fois par les canaux

communs et par des canaux particuliers. Les unités sont construites de façon à faciliter la maintenance, c'est-à-dire que chaque élément est caractérisé par ses éléments d'entrées et de sorties. La logique combinatoire permet de connaître l'élément en panne et dans ce cas, la puissance du système est diminuée mais non annulée. Les calculateurs de contrôle sont à même, par l'intermédiaire de programmes, de détecter les circuits défaillants.

Ils ont aussi la possibilité de se tester l'un l'autre pour rechercher les pannes pouvant les affecter. Un autre de leur rôle est l'initialisation du système.

3.8 - RESUME DES CARACTERISTIQUES DU SYSTEME

Le calculateur décrit plus haut rassemble les caractéristiques de base d'un multi-calculateur travaillant en multiprogrammation, à cause des moyens de communication intermodulaires et de l'organisation de la mémoire du calculateur. Les caractéristiques résumées ci-dessous donneront les avantages définis sur les calculateurs traditionnels..

- a) - Un haut degré de concurrence en fonctionnement est réalisé par le traitement multidimensionnel inhérent dans une structure modulaire. La capacité de traitement de données est élevée à un haut degré de parallélisme dans le système qui est atteint par les opérations concurrentes de plusieurs unités utilisées, soit sur un problème commun, soit sur des problèmes différents. La capacité extrême du système est le fonctionnement réellement simultané de toutes les unités sous le contrôle complet du programmeur.
- b) - Une grande sûreté, une économie, et une facilité d'extension sont offertes par l'utilisation multiplexée de plusieurs unités identiques. L'expansion du système requiert simplement l'adaptation de dispositifs standard de communication. Aucune unité simple ne peut causer une défaillance complète du système, mais au plus une

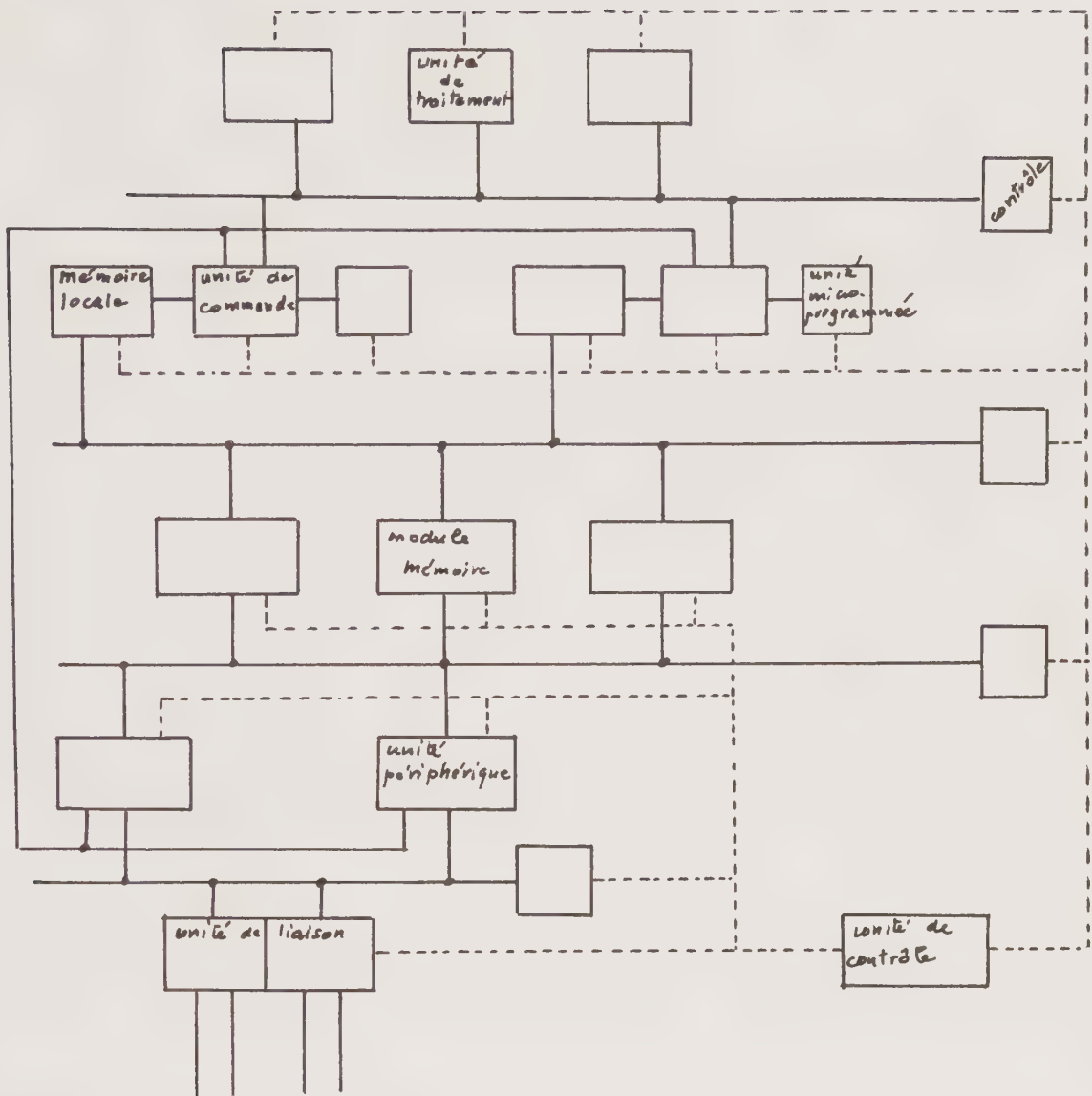


Fig 3-1

décroissance de sa capacité. Les unités en défaut peuvent être enlevées par la maintenance. L'utilisation du système à plein temps est par là assurée. Dans une telle organisation, les machines séparées peuvent être employées de toutes les façons désirées et quand une défaillance arrive, les interruptions de priorité interviennent et le programme entier de la machine en défaut est transféré à une autre par simple transfert de la mémoire. Cette opération ne demande que quelques microsecondes.

- c) - Les programmes sont protégés puisque seuls les calculateurs ayant le même numéro peuvent accéder aux blocs mémoire et dans les autres cas, la possibilité d'accès qui a été autorisée par le superviseur est vérifiée. Le secret de l'information est ainsi assuré.

3.9 - CONFIGURATION

La configuration du système la plus simple consiste en au moins une unité de contrôle (y compris l'arithmétique), deux unités périphériques (y compris l'unité de liaison canal), quelques bancs de mémoire et un calculateur de contrôle.

Dans le cas général, tous les schémas sont possibles à condition de respecter la logique du système, bien évidemment.

CHAPITRE IV

MECANISME D'ADRESSAGE, MEMOIRE PRINCIPALE, MEMOIRE LOCALE,

MEMOIRE ETENDUE

Le mécanisme d'adressage doit être tel qu'il soit à la fois simple et efficace. De sa simplicité dépend la rapidité des opérations et la diminution du risque d'erreur; de son efficacité, la rapidité d'accès aux informations à traiter. Le découpage de la mémoire, s'il est conçu en fonction de ce mécanisme, peut apporter beaucoup de souplesse et de facilité pour l'accès à cette mémoire.

4.1 - ADRESSAGE

4.1.1 - Choix de la structure

Deux modes de représentation sont possibles :

- mode octal (quantité d'information, le sextet soit 2 chiffres octaux),
- mode hexadécimal (quantité d'information, l'octet soit 2 chiffres hexadécimaux).

Plusieurs remarques s'imposent :

- la quantité d'information contenue dans un octet permet une plus grande latitude (256 possibilités contre 64),
- la tendance à la standardisation pousse à l'utilisation du code USASCII à 7 digits, ce qui implique la nécessité de 8 digits,

- il est souhaitable de pouvoir adresser la quantité d'information minimum pour travailler au niveau du caractère.

Pour ces raisons, nous adoptons la configuration hexadécimale, sans cependant cacher qu'il s'agit ici d'un choix pouvant être contro-versé.

4.1.2 - Choix de la longueur du mot machine

Le mot machine, compte tenu du désir d'adresser le caractère, contient un nombre de caractères égal à une puissance de 2. Le choix peut alors se porter sur :

- 1 octet (8 bits), quantité d'information minimale
- 2 octets (16 bits)
- 4 octets (32 bits)
- 8 octets (64 bits)

L'octet et le biocet représentent trop peu d'information. 32 bits donnent la possibilité en arithmétique fixe de nombres s'étendant jusqu'à 2^{32} , soit $4K^3$. Cependant, cette information peut ne pas être suffisante pour un traitement de données scientifiques et d'autre part, les arithmétiques flottantes ne donnent pas assez de précision avec seulement 32 bits. Nous choisirons un mot de 64 bits, soit 8 octets.

4.1.3 - Type d'adressage

Nous pensons que l'adressage sous la forme (base, déplacement) est le plus adapté aux travaux car il permet avec un nombre de bits limités d'atteindre des valeurs d'adresses très importantes.

Le choix de la valeur du déplacement est important.

- Un octet représente 256 possibilités ce qui est trop faible,
- 2 octets donnent 64K possibilités ce qui est trop grand, compte tenu de la segmentation des programmes.
 - 12 digits (4096) ou 13 digits (8192) correspondent à une taille intermédiaire suffisamment petite pour ne pas occuper trop de place et suffisamment grande pour réaliser un adressage aisé.

4.1.3.1 - Caractéristique d'adressage

Pour faciliter le travail, nous avons dit plus haut que l'octet devait être adressable. Cependant, la plupart des instructions travaillant au niveau du mot, il faut tenir compte de cet impératif. Nous avons aussi indiqué plus haut que l'adressage se faisait sous la forme base + index + déplacement. Le registre de base est cadré sur un mot machine. Suivant le type d'instruction, la somme (index + base) indique un déplacement en mot, 1/2 mot, 1/4 mot ou octet.

Une instruction mot cadre la somme index + base sur le mot.

Une instruction octet cadre la somme index + base sur l'adresse du mot contenant l'octet (décalage de la somme de 3 à droite), le reste indique la position de l'octet.

Exemple :

Soient les valeurs suivantes :

$$(X) + D = 24_{16}$$

$$B = 400_{16}$$

L'adressage du mot nous donne une adresse finale : 424

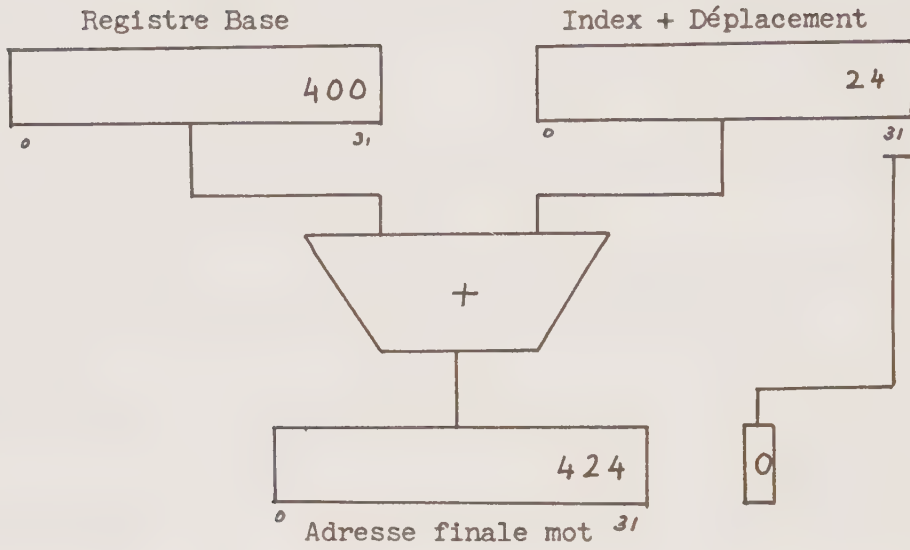


Fig 4-1

L'adressage de l'octet nous donne une adresse finale du mot : 404, avec un déplacement de 4 octets à l'intérieur du mot.

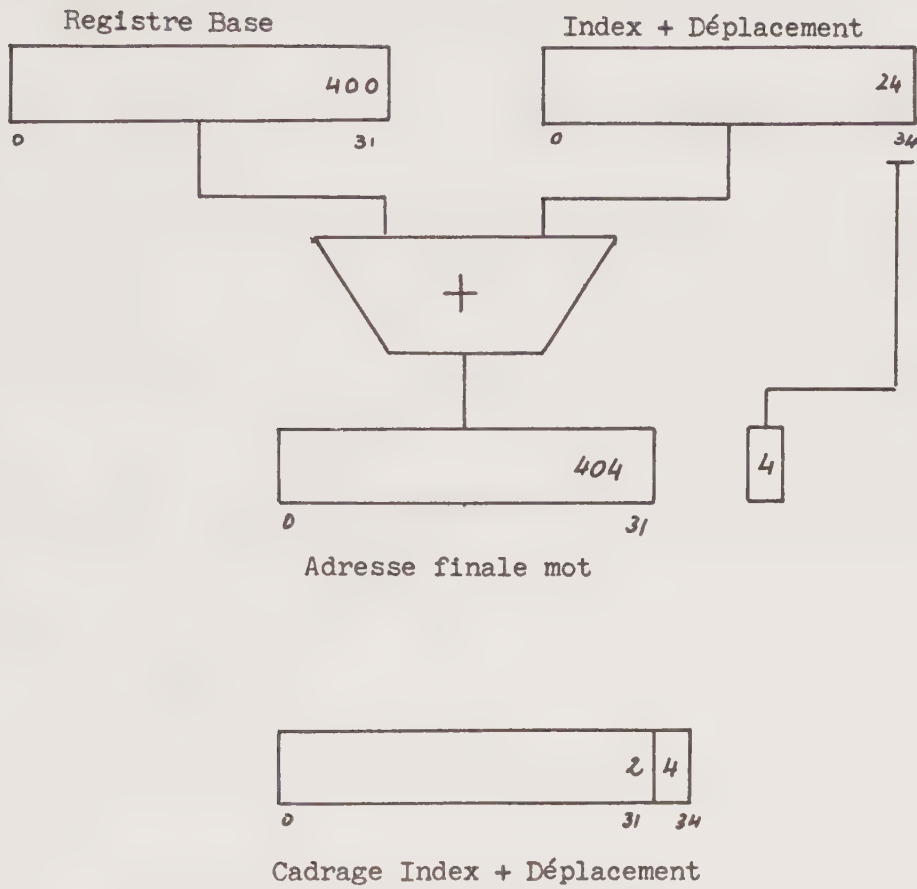


Fig 4-2

Cette méthode a un double avantage :

- 1°/ - La plupart des instructions s'effectuent sur un mot, cela nécessite moins de digits pour indiquer la même quantité d'information.
- 2°/ - L'indexation permet de travailler avec le même registre d'index sur des tableaux définis par des mots, des 1/2 mots ou des octets puisque le cadrage se fait automatiquement.

4.1.3.2 - Découpage logique

Nous appellerons chapitre un ensemble de 4 096 mots consécutifs.
(adresse $XX000_{16}$)

Nous appellerons page un ensemble de 256 mots consécutifs.
(adresse $XXX00_{16}$)

Nous appellerons ligne un ensemble de 16 mots consécutifs.
(adresse $XXXX0_{16}$)

L'adresse du chapitre est l'adresse du premier mot du chapitre et ainsi de suite.

Une adresse s'écrit donc sous la forme :

CH || P || L || D'

CH étant l'adresse du chapitre,

P l'adresse de page (4 digits),

L l'adresse de la ligne (4 digits),

D' l'adresse du mot dans la ligne (4 digits).

Ces définitions nous serviront plus loin.

4.2 - MEMOIRE PRINCIPALE

La nécessité de ne pas attribuer trop de mémoires ou trop peu de mémoires à un programme et la nécessité de protéger les programmes contre les accès intempestifs conduisent à une segmentation de la mémoire. Cette segmentation peut se faire de deux manières : par programmation ou par circuiterie.

La segmentation par programme est très lourde et ne peut de ce fait rendre de grands services dans le cas général.

La segmentation par circuiterie peut se faire à deux niveaux : au niveau de l'unité centrale ou au niveau de la mémoire.

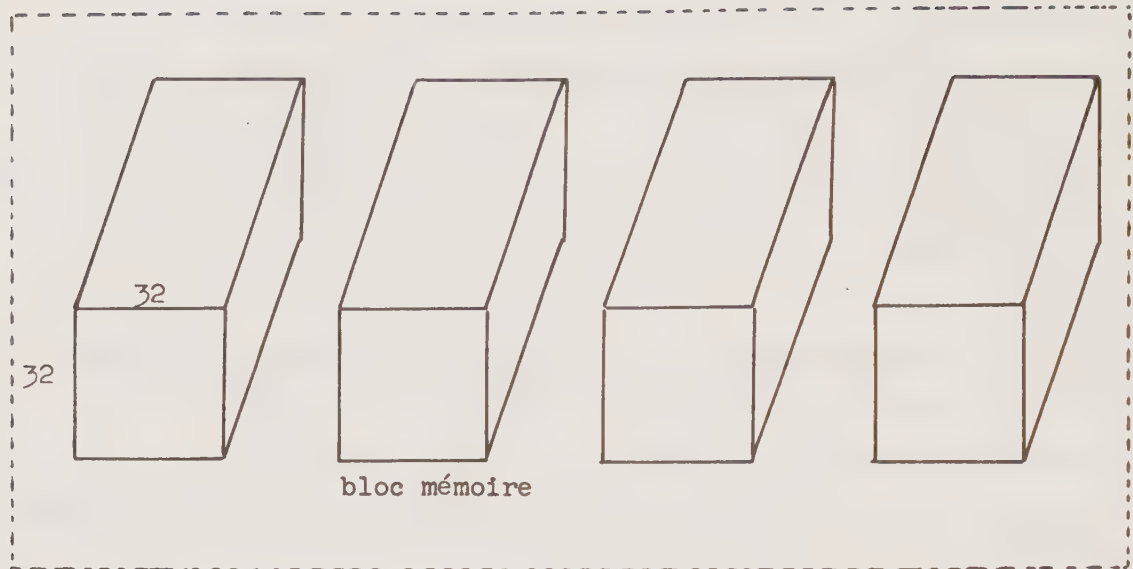
Dans le cas présent, on veut augmenter la rapidité et la sûreté du fonctionnement, donc la segmentation doit être simple et univoque. Alors les modules doivent être distingués par l'affichage d'un registre (modifiable par superviseur) qui indique quelle portion de la mémoire ils représentent.

4.2.1 - Module

Chaque module mémoire est composé de 16K mots.

Ce module possède un numéro logique et un numéro physique.

Le numéro logique permet l'attribution de modules par l'intermédiaire du superviseur à une unité de contrôle, par exemple le module 113 appartiendra à l'unité de contrôle 5 : le registre d'affectation contiendra la valeur 5. La possibilité de modification du numéro permet l'attribution d'un module libéré par une unité de contrôle à une autre unité de contrôle.



Module mémoire

Fig 4-3 a

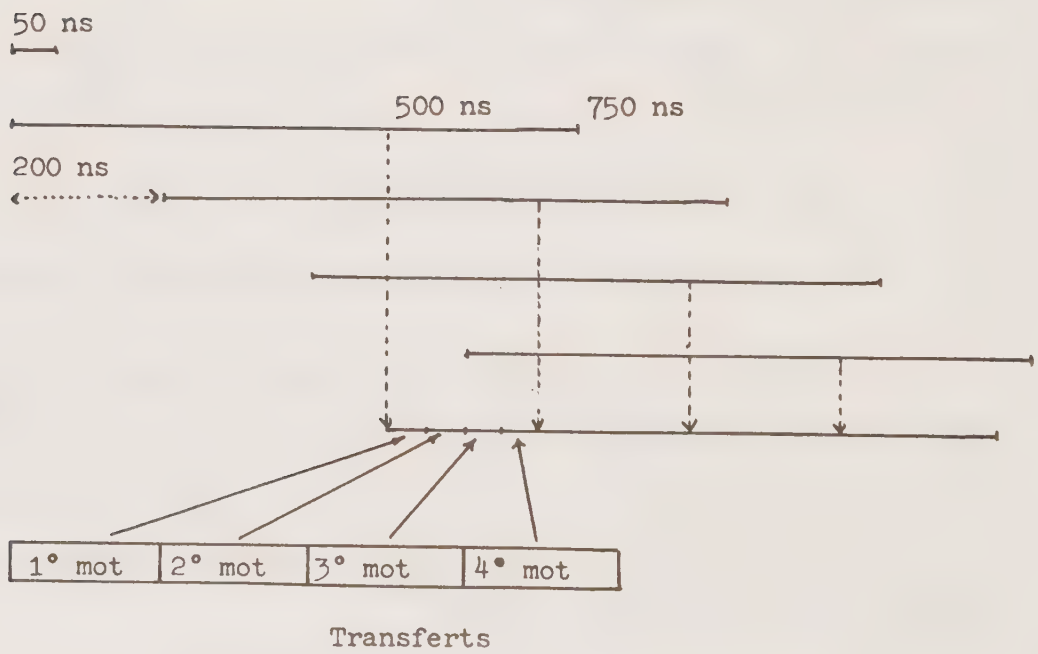


Fig 4-3 b

4.2.2 - Possibilités d'adressage

La possibilité d'adressage de la mémoire s'étend jusqu'à $16K^2$ (2^{24}) mots. Cependant, dans le cas général, un programme ne peut accéder qu'à une partie de la mémoire : nous en parlerons plus loin.

4.2.3 - Découpage physique d'un module

Chaque module est décomposé en 4 blocs de 4K mots.

Un bloc est une matrice 32×32 (K), de profondeur un mot de 32 octets (+ parité) soit 272 bits (un bit de parité pour 2 octets).

4 mots de 64 bits (1 mot de 272 bits) sont lus au cours d'un cycle d'écriture/lecture d'une durée de 15 cycles d'horloge (750 ns). Un registre pouvant contenir ces 272 bits est associé à ce bloc. Au cours d'un transfert d'enregistrement, les accès aux blocs sont déclenchés à des intervalles de 4 cycles d'horloge (200 ns), ce qui permet un transfert à une vitesse de 1 mot par cycle (voir figure).

4.2.4 - Découpage logique d'un module

Nous rappelons que nous appelons page un groupe de 256 mots consécutifs. Un module contient donc 64 pages.

A chaque page est associé un registre page qui contient le numéro de la page. En effet, nous avons voulu limiter les manipulations de données, aussi la page déterminée par un mot se situe dans n'importe quel module associé à la mémoire. La reconnaissance d'un code sélectionne la bonne page par l'intermédiaire des registres associatifs. La segmentation de la mémoire est alors automatique.

Cependant, il est possible par positionnement d'un drapeau de ne pas tenir compte de cette segmentation afin de permettre un travail plus rapide lorsque les pages sont ordonnées et le module entièrement affecté à un programme.

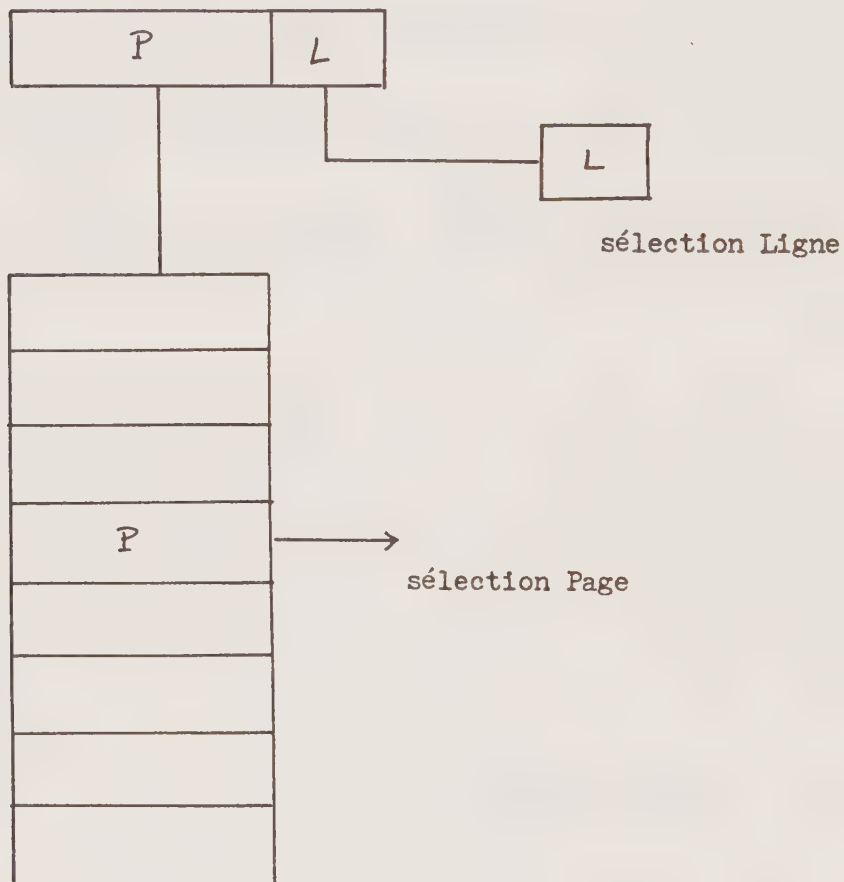


Fig 4-4

4.2.5 - Transfert

Pour faciliter et augmenter la vitesse des transferts, les registres lecture mémoire et écriture mémoire sont distincts, ce qui permet d'échanger des informations bloc à bloc d'un seul mouvement, par exemple transfert du contenu d'une page et remise à zéro simultanée. L'échange d'une page demande 500 ns (temps d'accès premier groupe de 4 mots) + 50 ns par mot, soit $(500 \text{ ns} + 50 \times 256) = 13,3 \mu\text{s}$. Une ligne est transférée en $(500 + 16 \times 50) = 1,3 \mu\text{s}$.

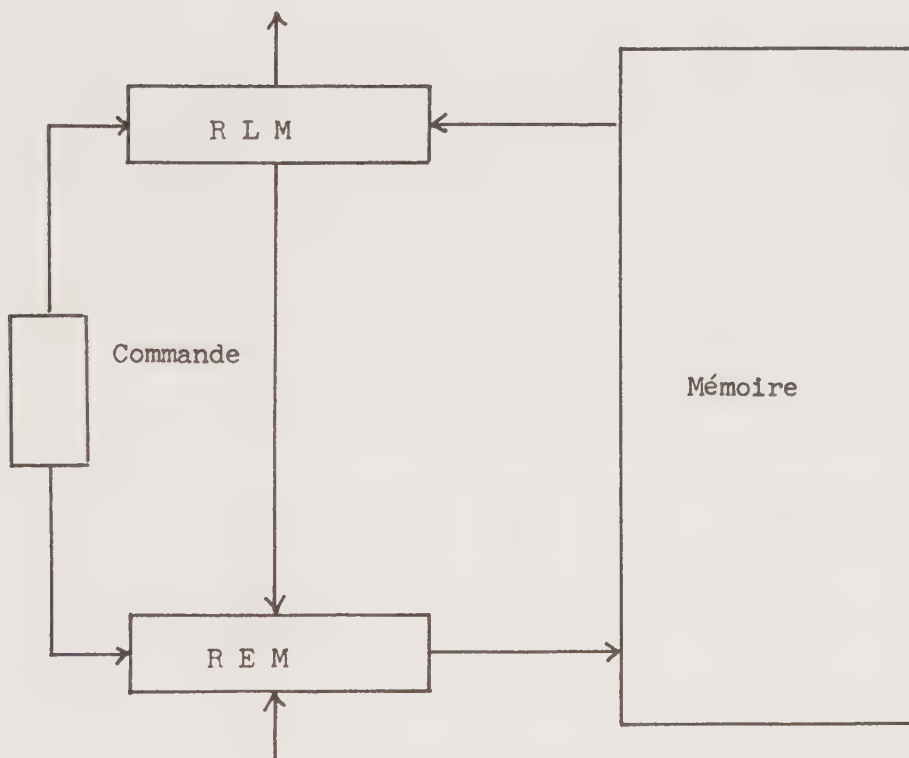


Fig 4-5

4.2.6 - Accès multiples

Il doit être possible pour plusieurs programmes d'accéder à un même bloc lorsque ce bloc contient des informations communes. Dans ce cas, l'autorisation d'accès interdit toute modification de l'état de la mémoire afin de ne pas perturber le fonctionnement par destruction des fichiers communs. Pour cela, la seule opération autorisée dans les fichiers partagés est la lecture. Si l'on veut opérer une fusion de données en provenance de deux programmes, cela peut se faire par concaténation des segments, c'est-à-dire que les numéros des pages seront réaffectés.

4.2.7 - Protection

Les programmes en exécution sont automatiquement protégés contre les accès intempestifs des programmes extérieurs puisque l'accès à une page ne se fait qu'à la condition que l'adresse du mot appartienne à l'unité de contrôle pilotant ce programme et au programme en cours. Pour cela, un champ de 6 bits est associé à l'adresse d'un mot en mémoire centrale. Les 3 premiers chiffres indiquent le numéro de l'unité de contrôle concernée et alertent les modules attachés à cette unité et ceux-là seulement.

A l'intérieur d'un module, les 3 chiffres suivants donnent le numéro du programme piloté par cette unité de contrôle. C'est-à-dire que la multiprogrammation se fait avec un nombre variable de tâches, ce nombre étant cependant inférieur à 8. Sous cette double condition, la protection est totale : en effet, ce champ de 6 bits est associé à l'adresse du mot par câblage au moment de la génération de la demande, et ne peut être modifié.

Tout programme a comme adresse de base l'adresse zéro. C'est-à-dire que le programmeur aura toujours l'impression d'occuper la mémoire à lui tout seul. Toute référence à une adresse absolue se fait à l'adresse absolue déterminée par le calcul avant adjonction du champ de protection.

A l'intérieur d'un programme, une page peut être :

- ouverte, c'est-à-dire que lecture, écriture et exécution sont permises
- protégée contre l'écriture, c'est-à-dire que le programme ne peut écrire dans la page,
- protégée en lecture, c'est-à-dire que le programme ne peut ni écrire ni exécuter les instructions dans la page,
- fermée : le programme ne peut qu'exécuter les instructions. Ce dernier cas permet l'exécution de programmes dont on veut éviter une copie par le programmeur. Lorsque la page est déclarée fermée, les instructions ne peuvent être lues que par l'unité de commande au cours du cycle d'instruction pour alimenter le flot d'instructions. Les références à des adresses internes ne se font que par des branchements.

4.3 - MEMOIRE LOCALE

Le concept de mémoire locale s'est imposé afin de minimiser les temps de réponse de l'unité de mémoire centrale aux demandes émises par le programme.

Deux tendances s'opposent :

- l'utilisation d'une mémoire très rapide de grande capacité alliée à une mémoire lente lui servant d'extension, les transferts se faisant à grande vitesse,
- l'utilisation d'une petite mémoire locale, représentation partielle de la mémoire centrale.

Les différentes études menées jusqu'ici ont montré que ces deux systèmes étaient très proches. Cependant, cela revient plus cher de fournir une grande mémoire très rapide et nous avons adopté la mémoire locale pour ces raisons et pour des raisons liées à la micro-programmation dont nous parlerons plus loin.

4.3.1 - Justification de la mémoire locale

L'utilité de la mémoire locale dans un système est liée à la réduction du nombre d'appels à la mémoire centrale et à la rapidité avec laquelle les informations déjà en mémoires locales sont obtenues. Le transfert par bloc est lié à ce besoin de rapidité.

Cependant, les caractéristiques de transfert ne sont pas intuitives et on doit déterminer :

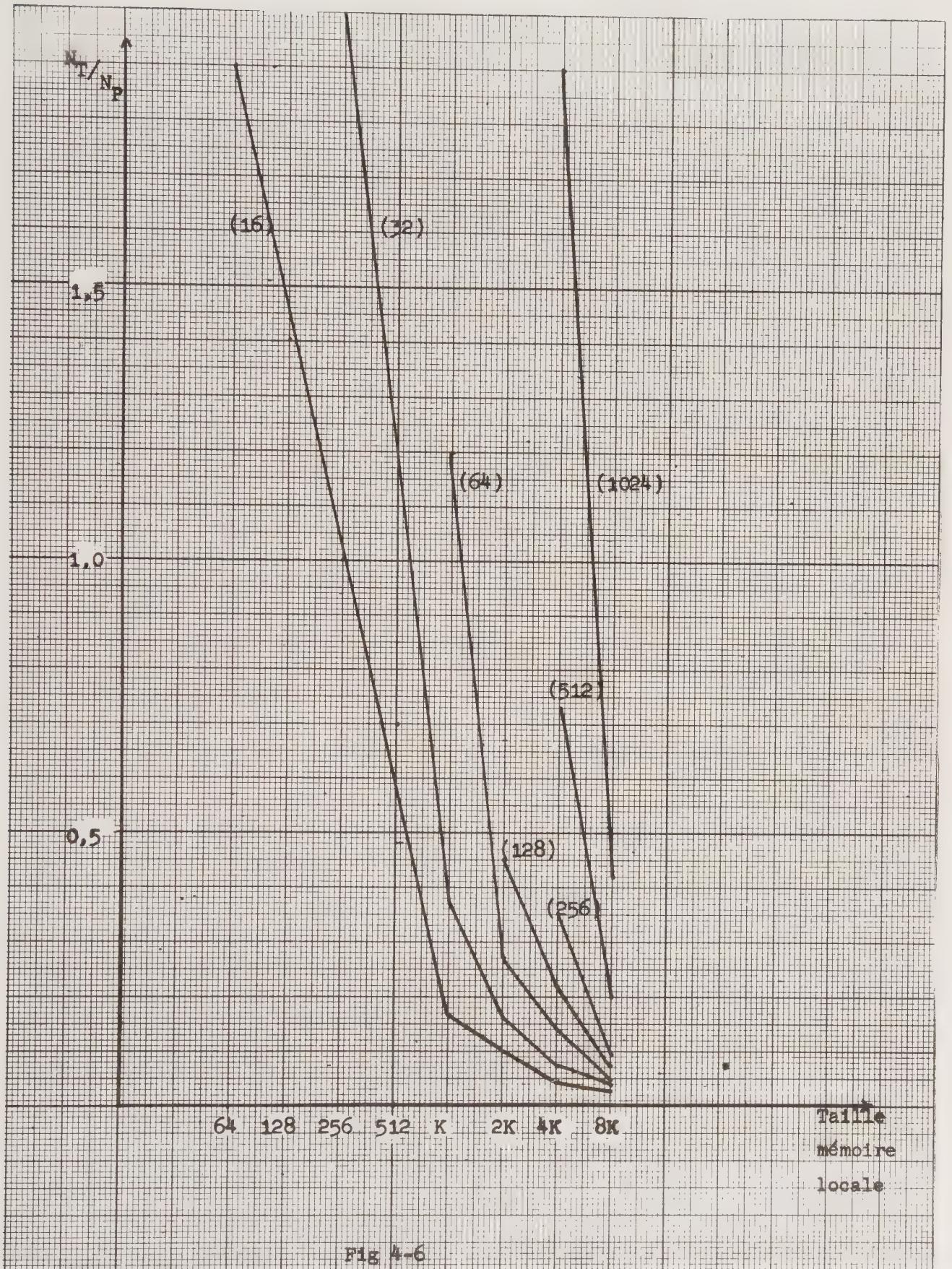
- le nombre de mots dans une ligne,
- le nombre de lignes en mémoire locale,
- l'algorithme de remplacement,
- la forme d'adressage.

La forme des références détermine la taille du bloc.

- L'adressage aléatoire donne une probabilité de ne pas trouver la référence en mémoire locale égale à $(1 - T_L/T_M)$

T_L étant la taille de la mémoire locale,

T_M la taille de la mémoire centrale.



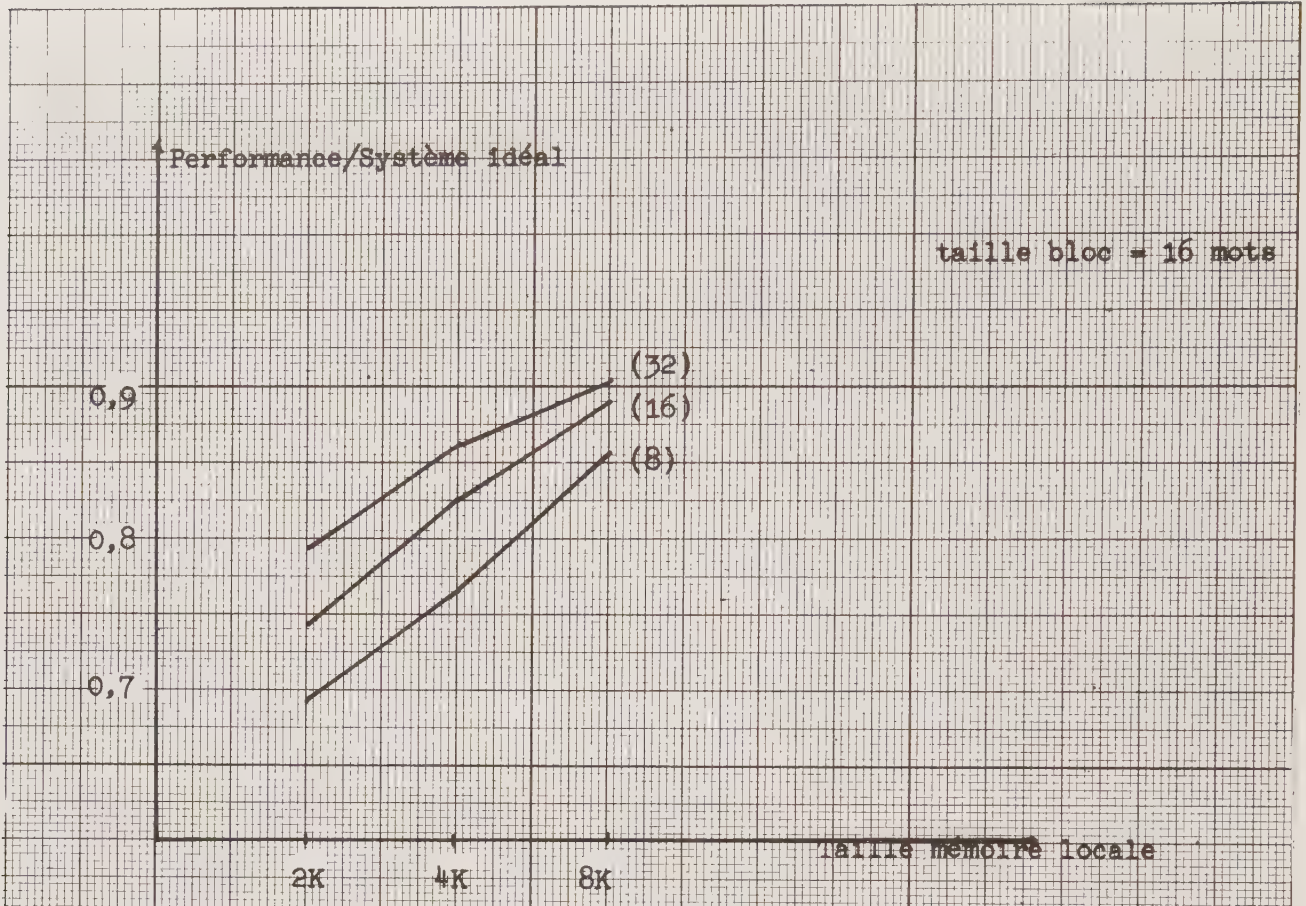


Fig 4-7

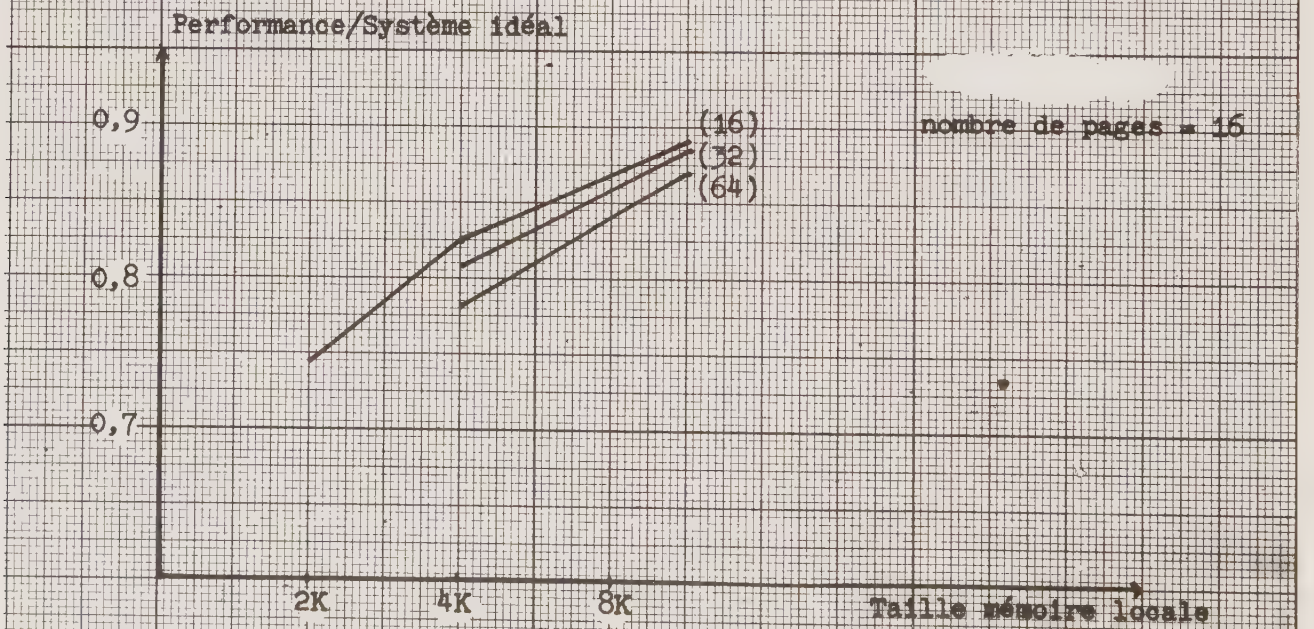


Fig 4-8

Dans ce cas, les références étant équiprobables, la taille du bloc ou la variation de l'algorithme n'ont pas d'influence sur les transferts.

- L'adressage séquentiel présuppose la succession des appels, la probabilité de ne pas trouver un mot est égale à $1/L_B$ où L_B est la longueur du bloc en mot.

Les différentes études menées jusqu'ici nous amènent à adopter le principe retenu pour le système 360/85.

La figure 4.6 donne les courbes exprimant le rapport entre le nombre de bits traités N_p par l'unité de commande et le nombre de bits transférés N_T en fonction de la taille de la mémoire locale et de la longueur du bloc.

Les figures 4.7 et 4.8 donnent les performances relatives à un système avec une mémoire locale de taille variable et :

1 - un nombre de pages variables et une taille de bloc égale à 16 mots,

2 - un bloc de tailles variables et un nombre de pages égal à 16.

Il aurait été possible de réaliser un système de multi-programmation analogue au "slot" du 6600. Cependant, dans ce cas, toute référence non satisfaite doit attendre un cycle complet (dans notre cas 400 ns) et tout programme en attente dégrade les performances du système puisque le temps alloué n'est pas utilisé.

4.3.2 - Description du principe

La mémoire locale n'est pas adressable par le programme mais est utilisée pour contenir les portions de mémoire centrale utilisées le plus souvent. La plupart des recherches de l'unité de commande se réfèrent à la mémoire locale. Si une information n'est pas contenue en mémoire locale, la portion contenant la donnée sera chargée et les données relatives à une autre portion abandonnées. Ce remplacement se fait automatiquement.

4.3.3 - Organisation de la mémoire locale

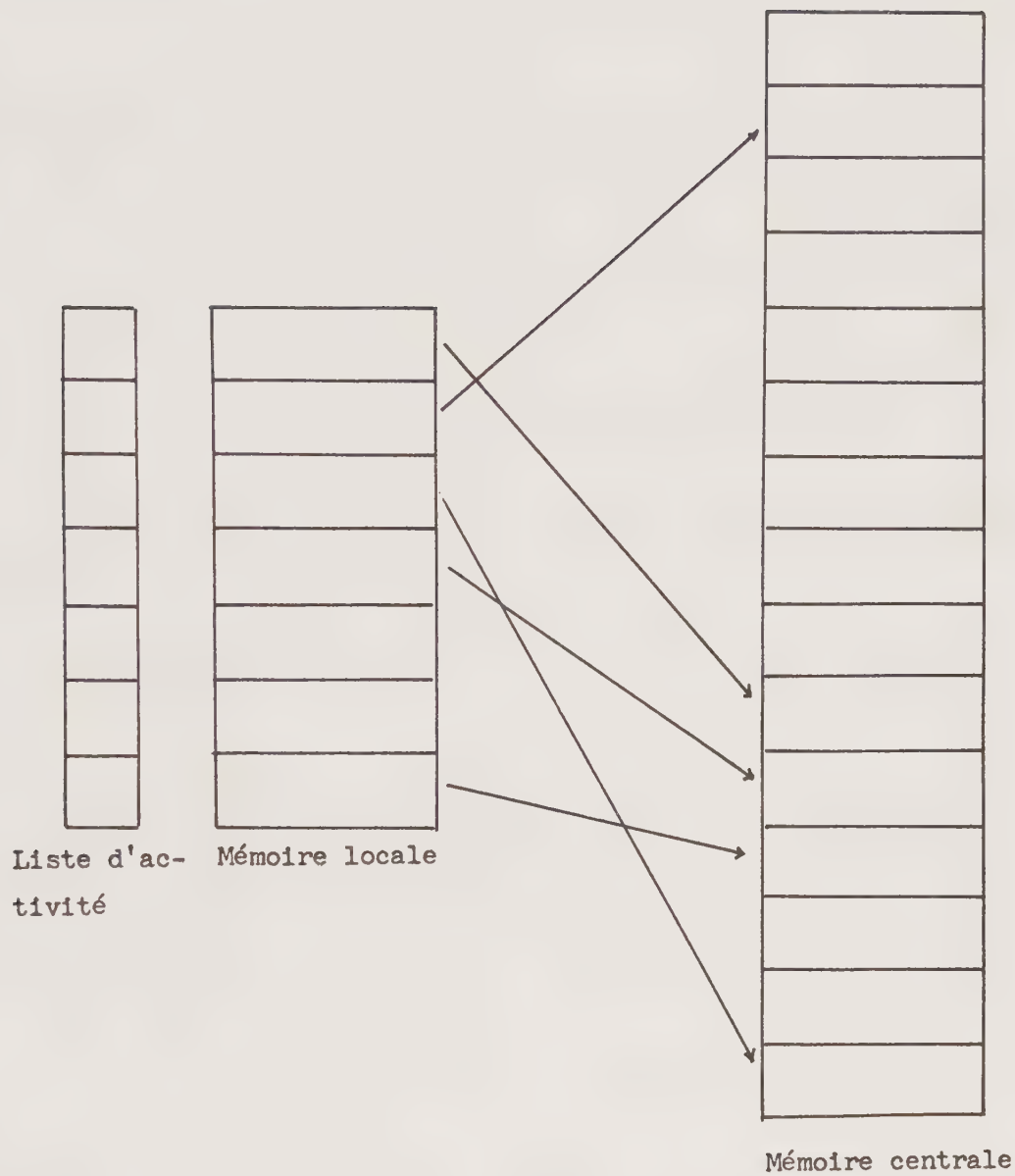
La mémoire locale est une mémoire de 4K mots dont le cycle d'écriture/lecture est de 100 ns.

La mémoire locale et la mémoire principale sont divisées logiquement en page de 256 mots consécutifs. Pendant les opérations, une correspondance est établie entre les pages de la mémoire locale et celle de l'unité centrale. Cependant, par suite de la limitation de la mémoire locale, la plupart des pages de la mémoire centrale ne sont pas référencées par une page en mémoire locale.

Chacune des pages de la mémoire locale a un registre d'adresse pointant sur la page de mémoire centrale à laquelle il est assigné.

4.3.3.1 - Affectation d'une page en mémoire locale

L'affectation des pages est effectuée dynamiquement de façon à représenter les pages couramment utilisées par le système. Si une recherche en mémoire centrale est effectuée à une page qui n'est pas référencée en mémoire locale, alors l'une des pages de la mémoire locale est réaffectée suivant une liste de priorités.



Affectation des pages

Fig 4-9

La page en tête est la page la plus récemment appelée et ainsi de suite. A chaque référence à une page en mémoire locale ou en mémoire centrale, cette page est positionnée en tête de liste par réaffectation du numéro de priorité. C'est la page de plus basse priorité qui est remplacée la première.

Cependant, à chaque référence, toute la page n'est pas chargée mais seulement la ligne contenant le mot désiré. Lorsque cela est requis, les lignes restantes sont chargées et un bit de validité positionné indiquant la présence de la ligne en mémoire locale.

4.3.3.2 - Mises en garde

Les rangements affectent toujours la mémoire centrale.

Si ce rangement affecte une portion déjà en mémoire locale, celle-ci est aussi modifiée sans cependant changer la liste de priorité, provoquer le chargement d'une ligne ou le remplacement d'une page.

Puisque la page en mémoire locale est une copie fidèle de celle en mémoire centrale, toute réaffectation n'implique que le chargement de la ligne concernée et la modification de la liste de priorité et du registre d'adresse. Les informations de remise en garde peuvent être stockées de façon à ne pas bloquer le système si les rangements ne peuvent être effectués immédiatement.

Trois cycles (150 ns) sont nécessaires pour rechercher une donnée en mémoire locale, le premier pour reconnaître la validité de la recherche, les deux autres pour l'acquisition. Toutefois, la mémoire locale étant entrelacée, une demande par cycle peut être effectuée. Cependant, si la donnée n'est pas présente en mémoire locale, un temps plus long est requis pour accéder à la donnée.

Nous avons indiqué que les mots de 64 bits sont groupés par 4 dans la mémoire principale. La largeur de la voie de transfert est de 64 bits. Les transferts se font à raison de 1 mot par cycle de 50 ns.

Cependant, pour accroître les performances, le groupe de 4 mots dans lequel la référence se trouve est adressé le premier à la mémoire locale et à l'unité de commande de façon à accélérer les opérations.

Les entrées-sorties se font en mémoire principale : cependant, une modification de la mémoire principale ne peut être faite que lorsque le programme n'est pas en exécution, c'est-à-dire alors que la mémoire locale n'est pas occupée par ce programme.

Nous expliquerons au chapitre VI l'utilisation de la mémoire locale en microprogrammation.

4.4 - MEMOIRE ETENDUE

Il est possible d'adjoindre au système une mémoire étendue de grande capacité accessible par lecture ou écriture. Cette mémoire permet une affectation supplémentaire par multiple de 4K mots afin d'augmenter la capacité de la mémoire centrale.

4.4.1 - Modules

Elle est composée de modules de 64K mots groupés par blocs de 8K mots. Le temps d'écriture/lecture d'un mot mémoire étendue est de 6,4 μ s, soit 8 mots de 64 bits par cycle. La mémoire étant entrelacée 8 fois, cela permet de transférer un mot en 2 cycles mineurs.

A chaque bloc est associé un registre de 544 bits qui contient un mot mémoire étendue (parité comprise). A chaque chapitre (4K mots) est associé un registre indiquant à quelle unité il appartient et son adresse.

4.4.2 - Adressage

L'adressage se fait de manière identique à celui de la mémoire centrale. Toutefois, les seules opérations possibles sont la lecture et l'écriture. La commande se fait par l'intermédiaire de deux instructions non privilégiées.

Le mot d'état programme contient le nombre de chapitres maximum. Un programme peut adresser au plus $16K^2$ mots, soit 4K chapitres.

Les registres sont modifiables par le superviseur et un chapitre peut être protégé contre l'écriture.

CHAPITRE V

UNITE DE COMMANDE, DE TRAVAIL, D'EXECUTION

L'exécution d'un programme demande la mise en oeuvre de nombreux composants du système, tant pour le décodage que pour l'exécution des instructions. Les dispositifs doivent être les plus efficaces possibles pour minimiser le temps d'exécution. Cependant, ils doivent aussi être sûrs afin de ne pas provoquer de défaillances du système. La solution envisagée ici est de modulariser au maximum les différentes parties de l'unité centrale classique sous forme de boîtes noires assurant différentes fonctions.

5.1 - UNITE DE COMMANDE

L'unité de commande assure le contrôle et la répartition des informations entre les différentes unités de travail (unité de traitement, mémoire). Elle assure le décodage des instructions et sélectionne les lignes d'accès. En effet, seul un décodage partiel est effectué au niveau de l'unité car une partie des instructions n'est décodée qu'au niveau de l'unité de traitement ; par exemple, pour un travail destiné à l'unité arithmétique fixe, les signaux codés sont envoyés vers cette unité qui assure les opérations. Ce qui revient à dire que chaque module simple peut être considéré, grossièrement, comme un mini-calculateur très spécialisé.

Une partie seulement des instructions est câblée. En effet, si la microprogrammation permet un gain appréciable dans le domaine de la souplesse, elle entraîne cependant une lourdeur dans la gestion

des signaux. Aussi, les instructions les plus courantes (lecture, écriture) sont câblées : la liaison entre un microprogramme et ces instructions se fait de manière aisée, le code de commande étant le même pour la microprogrammation.

5.1.1 - Nombre de programmes en multiprogrammation

Nous avons dit au chapitre III que le nombre de programmes en multiprogrammation pouvait s'étendre jusqu'à 8. Ce nombre n'est pas complètement arbitraire. En effet, pour des raisons de commodité, c'est une puissance de 2 (2, 4, 8, 16). Cependant, 4 programmes en multiprogrammation, ce n'est pas suffisant pour des travaux courants et 16 programmes sont une charge trop importante de gestion pour une seule unité de commande. D'autre part, si nous tenons compte de la taille mémoire ($16K^2$ soit 2^{24}), nous voyons que chaque programme pourra accéder à 2^{18} mots. L'adressage absolu permet de concaténer les champs d'adressage et de contrôle pour adresser toute la mémoire.

5.1.2 - Caractéristiques de la multiprogrammation

Il existe deux modes de multiprogrammation :

- le premier dit mode normal dans lequel la commutation sur les différents programmes se fait de façon cyclique lorsque les conditions d'exécution le permettent, c'est-à-dire lorsqu'un programme est en attente d'exécution. Dès que les conditions d'exécution sont remplies, la commande est donnée au programme : par exemple, un programme ayant demandé une entrée-sortie et celle-ci étant terminée de façon satisfaisante, le programme en attente reçoit la commande si un autre programme est interrompu pour une cause quelconque.

- le deuxième mode est dit mode commandé ; la commutation est commandée par un autre programme qui joue alors le rôle de superviseur.

5.1.3 - Caractéristiques internes de l'unité de commande

Les circuits internes de l'unité de commande permettent :

- l'accès aux registres et mémoires de travail,
- l'accès à la mémoire générale,
- l'accès à la mémoire locale,
- l'accès à l'unité microprogrammée,
- la commande des unités de traitement.

L'unité de commande génère ses commandes toutes les 50 ns vers les différentes unités. Son rôle est analogue à celui d'un centre de distribution, qui assure un flot constant et correct de marchandises du producteur au consommateur en satisfaisant aux priorités et demandes émises par les différents participants.

Plusieurs cas peuvent se présenter :

- tous les programmes sont exécutables ou une partie seulement : la commande est donnée au programme le plus prioritaire,
- aucun programme n'est exécutable : l'unité demande à être testée par l'unité de contrôle centrale,
- lorsqu'un programme en exécution est interrompu soit par signaux externes, soit par signaux internes, l'unité de commande donne l'exécution au plus prioritaire.

Pour respecter l'ordre chronologique, un indice est affecté à chaque commande. Si un blocage intervient au niveau d'une commande, la reprise a lieu à cette commande et provoque l'annulation de toutes les commandes suivantes. Si ce blocage a des causes externes, (par exemple non disponibilité d'une unité), un signal est généré afin de provoquer une commutation automatique.

5.2 - UNITE LOCALE DE TRAVAIL

L'unité locale de travail est l'unité contenant les informations nécessaires à l'exécution d'un programme et stockant les informations en provenance du superviseur, de l'unité de commande et des unités de traitement. Cette unité contient les registres de travail, des mémoires de travail et des indicateurs d'état concernant le déroulement du programme (condition de fonctionnement, étape, limite, etc...).

Chaque programme a à sa disposition une zone renfermant tous les renseignements (nombre de zones égal à 8) pour permettre une commutation rapide sans recherche en mémoire centrale.

5.2.1 - Registres de travail

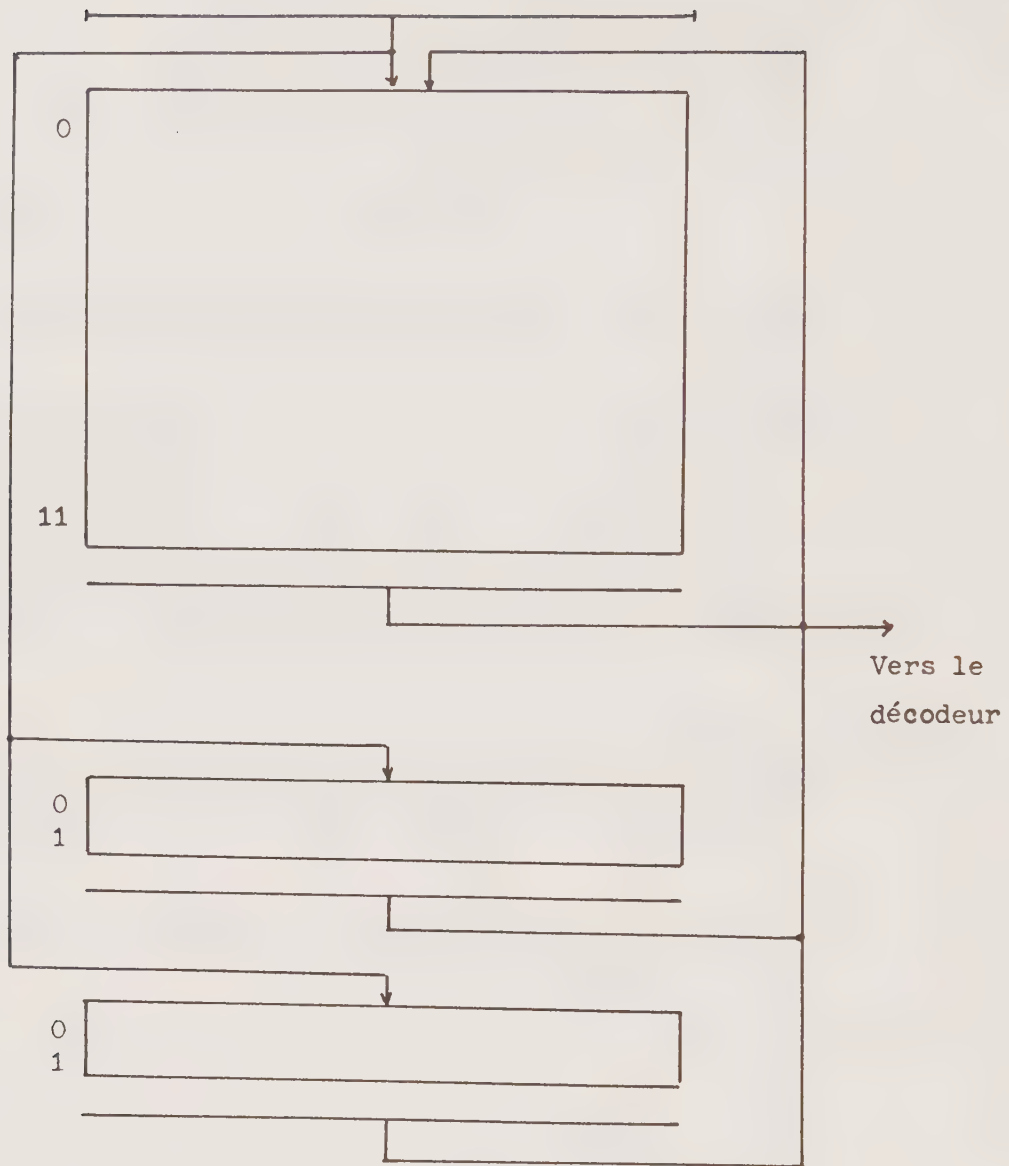
Un programme a à sa disposition un certain nombre de registres.

- Des registres de bases servant pour l'adressage, au nombre de 7 (B1 à B7), et ayant 24 digits,

- Des registres d'index servant pour les décalages, au nombre de 7 (X1 à X7), contenant 32 digits,

- Des registres de travail pour l'arithmétique fixe et la logique, au nombre de 8 (R0 à R7), contenant 64 digits,

- Des registres pour l'arithmétique flottante, au nombre de 4 (F0 à F3), contenant 64 digits chacun.



Pile instructions et tampons de branchement

Fig 5-2

Ces registres sont contenus dans la mémoire rapide de 32 mots (fig. 5.1) (soit 256 mots pour la totalité des programmes possibles). Cette zone n'est pas adressable par programme mais est accessible par le superviseur.

5.2.2 - Pile instruction

Puisque l'unité de commande est capable de générer un ordre toutes les 50 ns vers les unités de traitement, une instruction par cycle doit pouvoir être traitée.

Cependant, un important problème est celui des branchements qui rompent la séquence normale d'instructions.

Des mesures sur différents types de machines ont démontré que la part des branchements s'étend de 10 à 33 % des instructions exécutées, que la plupart (40 %) des branchements s'effectuaient en moins de 16 instructions, donc qu'une pile comprenant 16 instructions a 40 % de chance de contenir l'instruction où s'effectue le branchement.

De même, le problème des boucles qui sont fréquemment très courtes est important.

Pour cela, nous utiliserons une pile d'instructions contenant 12 mots pour alimenter le flot normal d'instructions et 2 piles de 2 mots pour les branchements.

5.2.2.1 - Recherche des instructions séquentielles

Une chaîne séquentielle d'instructions alimente continuellement l'unité instruction 4 mots par 4 mots.

Lors de l'initialisation du programme, 12 mots sont fournis d'un seul coup puisque la pile ne contenait aucune instruction. Un problème se pose lorsque les instructions recherchées sont à la limite de la zone affectée, car il peut y avoir débordement de la recherche. Pour lever ce cas pendant la période de recherche, l'adresse du bloc du mot généré est continuellement vérifiée et lorsque la limite est atteinte, la pile est remplie avec une instruction non opération. Si la pile ne contient que des instructions non opération, une interruption se produit signalant le débordement.

5.2.2.2 - Manipulation des branchements

La question de savoir si un branchement est effectué ou non est important puisque cela peut amener une rupture de séquence et rendre vain la recherche des mots suivant le branchement, surtout dans le cas des branchements courts. Dans le cas de la détection d'un branchement, 4 mots sont recherchés à partir de l'adresse de branchement. Pour faciliter l'adressage, un branchement ne peut se faire qu'à une adresse de mot. Cette recherche est faite en parallèle avec le flot normal. Si le branchement est fait, alors les registres fournissent les 4 premiers mots de la pile instruction.

Si le branchement est effectué en arrière à moins de 12 mots, on considère que le branchement sera prioritaire : dans ce cas, l'unité instruction se met en mode bouclé et arrête la recherche des instructions le long de la séquence. Les instructions suivant le branchement sont alors placées dans les deux registres de branchement. Puisque toutes les instructions sont dans la pile, la recherche des opérandes peut être accélérée puisque la ligne d'accès est totalement libérée.

5.2.2.3 - Interruptions

Les interruptions sont un cas particulier des branchements.

En effet, plusieurs cas peuvent se présenter :

- l'interruption arrête le programme (commutation) et les renseignements deviennent nuls,
- l'interruption n'arrête pas le programme et consiste en une seule opération : dans ce cas, cette opération est forcée dans le flot sans l'interrompre,
- l'interruption provoque le transfert à un autre point du programme : elle est alors traitée comme un branchement normal.

Pour conserver la possibilité de détecter une interruption précise, la séquence stricte d'instructions est conservée, autrement dit aucune opération ne pourra se terminer avant la précédente. Pour cela son exécution sera retardée dans le cas où elle risque de provoquer le même genre de défaut. Si les opérations successives s'adressent à des unités distinctes, alors le recouvrement pourra avoir lieu.

5.3 - UNITE D'EXECUTION

A chaque instant dans un calculateur, seule une partie de l'équipement est utilisée si l'on veut séparer les fonctions : aussi notre idée a-t-elle été de constituer un groupe de fonctions diverses, ce groupe étant géré par une unité spéciale pour chacune des fonctions.

Lorsqu'une demande de service adressée par l'une des unités est reconnue par le système de distribution, celui-ci la prend en compte dans une pile en conservant évidemment tous les renseignements concernant l'origine de la demande. Lorsqu'une des unités spécifiques est libre, il lui donne le travail. Un drapeau d'activité indique l'état de chaque unité et un indice de l'unité d'origine de façon à enchaîner le travail si plusieurs informations venant de la même unité sont en attente.

5.3.1 - Fonctions

Il existe plusieurs types de modules pour exécuter les fonctions du système.

- module arithmétique fixe (addition),
- module arithmétique fixe (multiplication),
- module arithmétique fixe (division),
- module logique,
- module décalage,
- module incrément (plus ou moins un),
- module arithmétique flottante (addition),
- module arithmétique flottante (multiplication/division),
- module traitement arithmétique décimal,
- module traitement caractères.

Ces modules possèdent chacun plusieurs fonctions, toutes identiques : la panne d'une fonction ne bloque pas le système mais au plus diminue la rapidité de fonctionnement.

Pour diminuer le risque de panne du distributeur, celui-ci est doublé et fonctionne en bascule.

5.3.2 - Traitements parallèles

Dans certains cas, il peut être intéressant d'effectuer un traitement parallèle d'opérations ; cela est possible puisqu'il existe plusieurs unités de même type et il est aussi possible de déclencher par le même signal de cadencement les opérations de façon à avoir un parallélisme total. Ceci cependant ne peut être fait que sous contrôle du superviseur. Le chaînage est indiqué dans le mot d'état programme.

5.3.3 - Arithmétique

Deux moyens sont envisageables pour réaliser l'arithmétique : soit l'arithmétique en complément à un, soit l'arithmétique en complément à deux.

La première a l'avantage de la rapidité et de la simplicité de l'électronique. Cependant, du point de vue du programmeur, cela entraîne l'existence de 2 zéros : $+0$ et -0 .

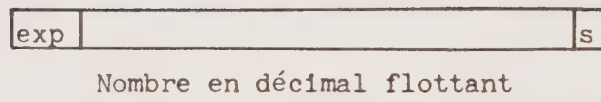
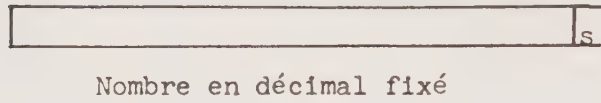
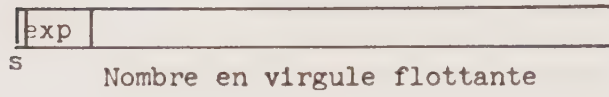
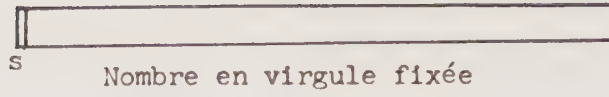
La seconde est plus lente puisque la complémentation demande l'incrément du plus bas poids, ce qui peut se répercuter sur le plus haut poids d'un nombre.

Nous adopterons la deuxième solution car cela facilite le travail du programmeur.

. En arithmétique fixe, les nombres sont représentés sous la forme complément à 2, signe en tête.

. En arithmétique flottante, les nombres sont représentés en complément à 2, signe en tête, l'exposant représentant des puissances de 16 étant augmenté d'un préfixe de façon qu'il soit toujours positif.

Certaines instructions permettent de passer aisément du format fixe au format flottant et réciproquement.



Signe décimal	Valeur	Fraction	Exposant
	A	+	-
	B	-	-
	C	+	+
	D	-	+
	E	+	+
	F	+	+

Fig 5-3

. L'arithmétique décimale se fait sur un registre qui contient 15 chiffres décimaux et le signe pour l'arithmétique fixe et 13 chiffres décimaux, 2 pour l'exposant et le signe pour l'arithmétique flottante.

CHAPITRE VI

MACHINES SEQUENTIELLES ET MICROPROGRAMMATION

6.1 - MACHINES SEQUENTIELLES ET SIGNAUX D'ETAT

Chacun des modules d'un système peut être regardé comme une machine séquentielle possédant un ensemble E de signaux d'entrées, S de signaux de sorties, et H d'états internes.

De façon plus précise, l'état du système et les signaux d'entrée définissent de façon unique les signaux de sortie et l'état que prendra l'état de sortie.

Ce système sera dit synchrone, s'il possède e entrées, s sorties et s'il est muni d'une source de synchronisation.

Les variables d'entrées et de sorties ne sont alors mesurées qu'à des instants privilégiés déterminés par le signal d'horloge. Ces instants ne sont pas forcément équidistants dans le temps, mais impliquent seulement un cadencement. Dans ce cas, la variable n'est pas le temps écoulé depuis le premier signal mais le rang du signal.

6.1.1 - Machines de Moore et Mealy

On distingue deux types de machines séquentielles selon que le signal de sortie ne dépend que de l'état machine (Moore) ou de l'état machine et du signal d'entrée (Mealy).

Dans le cas de la machine de Mealy, le symbole de sortie n'apparaît que pendant les transitions, dans l'autre cas (Moore) le signal de sortie persiste jusqu'au prochain signal.

6.1.1.1 - Machine de Moore

Une machine de Moore peut être définie par les relations :

$$h_{j+1} = F(e_1, h_j)$$

$$s_K = G(h_j)$$

où F est une application de l'ensemble $\{ E \times H \}$ dans $\{ H \}$

et G est une application de $\{ H \}$ dans $\{ S \}$.

6.1.1.2 - Machine de Mealy

Une machine de Mealy peut être définie par les relations :

$$h_{j+1} = F(e_1, h_j)$$

$$s_K = G(e_1, h_j)$$

F étant une application de $\{ E \times H \}$ dans $\{ H \}$

G étant une application de $\{ E \times H \}$ dans $\{ S \}$.

La différence entre Moore et Mealy réside dans la définition de G .

6.1.2 - Codage des signaux

Un système séquentiel synchrone est complètement déterminé lorsqu'on connaît les deux fonctions :

$$h = f(e, h) \quad (1)$$

$$s = g(e, h) \quad (2)$$

(2) restreint à $s = g(h)$ (3) pour Moore.

Ces deux fonctions sont complètement définies par les tableaux de transition ou le graphe associé.

Si nous considérons un système séquentiel ayant i symboles d'entrée, j symboles d'état et K symboles de sortie :

- les i symboles d'entrée pourront être codés en binaire au moyen d'une suite de ξ symboles, $2^{\xi-1} < i \leq 2^\xi$
- les j symboles d'état pourront de même être codés par ψ chiffres binaires, $2^{\psi-1} < j \leq 2^\psi$
- les K symboles de sortie pourront être également codés au moyen d'une suite de ω symboles, $2^{\omega-1} < K \leq 2^\omega$

On aboutit à un système ayant ξ entrées, ψ états, ω sorties.

L'équation (1) peut se décomposer en ψ fonctions booléennes de $\xi + \psi$ variables et (3) en ω fonctions booléennes de ψ variables.

6.1.3 - Extension vers un calculateur

Chaque fonction d'un calculateur peut être considérée comme un automate, l'ensemble des éléments constituant la circuiterie du calculateur.

Le codage des signaux prend alors une grande importance car les signaux de sortie d'un élément peuvent être des signaux d'entrée pour un autre comme l'indique la figure ci-dessous.

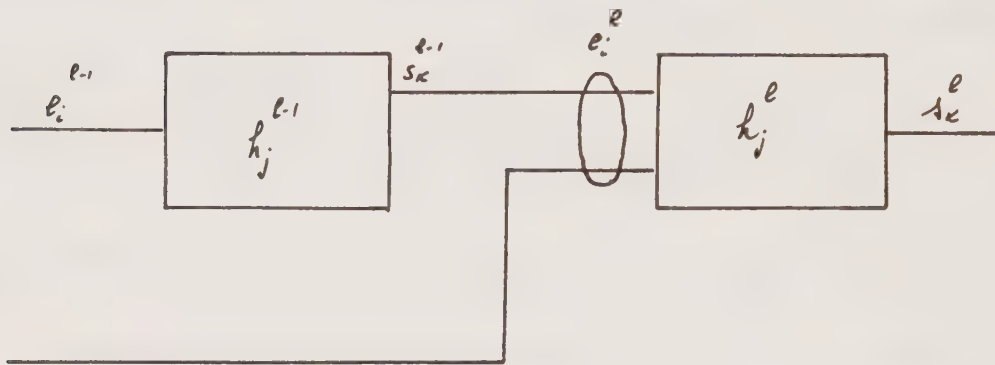


Fig 6-1

Nous avons alors :

$$e_i^l \in E^l \subset E$$

ensemble des signaux d'entrées,

$$s_k^l \in S^l \subset \Sigma$$

ensemble des signaux de sorties,

$$h_j^l \in H^l \subset \Theta$$

ensemble des signaux d'état.

6.2 - MICROPROGRAMMATION

Pour permettre une exécution rapide des instructions, il est plus normal de les câbler. Cependant, il est parfois difficile de câbler certaines instructions non parce que cela est impossible, mais pour des raisons d'économie ou parce qu'il peut être intéressant de les modifier.

Plusieurs façons de concevoir la microprogrammation s'affrontent.

La première, celle de WILKES, consiste à donner à l'ensemble des circuits de contrôle une structure comparable à celle du calculateur, c'est-à-dire comportant un registre de micro-ordres, un décodeur, et des circuits d'exécution, les micro-instructions étant générées les unes après les autres par les circuits logiques.

La seconde consiste à utiliser une mémoire non destructible, où les micro-instructions sont rangées de façon tout à fait semblable aux instructions d'un programme.

La troisième consiste dans des instructions sans adresse qui désignent certaines opérations ou conditions par le positionnement de 1 en un endroit précis de l'instruction et permettent l'utilisation de plusieurs de ces opérations ou tests simultanément, par exemple instruction d'échange registres du SDS 930.

Selon le type de travail, tel ou tel schéma est possible, mais il est clair que :

- la microprogrammation doit permettre au programmeur de tailler ses propres instructions d'une manière relativement aisée,
- le microprogramme est fixé et accessible soit par un code opération, soit comme microprogramme emboîté.

Nous pensons que les deux options ne sont pas incompatibles à condition de bien délimiter le champ des micro-instructions.

6.2.1 - Mémoires de microprogrammes

Chaque unité de contrôle a accès à une mémoire microprogrammée fixe de 4 096 mots de 64 bits.

Cette mémoire a un temps d'accès de 100 ns identique à celui de la mémoire locale.

Les microprogrammes enregistrés ne sont modifiables que par échange de modules physiques, ce qui nécessite l'arrêt du calculateur.

Cependant, comme il peut être intéressant pour un programmeur d'écrire ses propres microprogrammes, nous avons envisagé ce cas de la manière suivante : une instruction particulière appartenant au jeu d'instructions standard permet l'appel de micro-instructions dont l'implantation est effectuée en mémoire locale au moment du chargement avec un maximum de 4 pages.

Dans ce cas, un drapeau spécial annule l'affectation de la page considérée à une page mémoire.

Les pages contenant les microprogrammes ont obligatoirement les adresses de page les plus basses et ont la plus haute priorité d'accès. Lors d'une commutation, le mot d'état programme indique la présence de micro-instructions et le nombre de pages concernées. Le premier travail est alors de les implanter en mémoire locale.

Ensuite, si le contrôle est donné à un microprogramme résidant en mémoire locale, tout se passe comme si les micro-instructions étaient dans la mémoire microprogrammée.

6.2.2 - Représentation des micro-instructions

Dans le système, les différentes unités sont indépendantes, c'est pourquoi la microprogrammation présente des difficultés.

En effet, la commande directe des circuits implique une liaison permanente. Puisqu'en principe, les modules sont indépendants, il en résulte que le décodage de la micro-instruction ne peut se faire complètement au niveau de l'unité microprogrammée.

Lors de la lecture d'une micro-instruction, une partie seulement est décodée au niveau de l'unité de commande, ce qui permet de connaître quelle unité est concernée par la commande.

Ensuite la seconde partie est envoyée vers l'unité concernée et dans ce cas, il est possible d'envisager les deux types de microprogrammation décrits plus haut à savoir la commande par micro-ordres ou le circuit séquentiel.

Le circuit séquentiel présente des avantages au point de vue du temps puisque le cadencement est automatique et que l'on gagne le temps de décodage et de génération de l'instruction suivante.

Il doit être possible d'envisager le remplacement d'un module commandé par signaux séquentiels par un module commandé par micro-ordres sans changer la codification des micro-instructions. C'est-à-dire que les champs des micro-instructions doivent correspondre de façon biunivoque aux signaux de commande d'un circuit séquentiel. Cette dualité permet ainsi une totale ignorance des équipements par le rédacteur de micro-instructions.

Au niveau de chaque unité de traitement, il existe donc un décodeur pour effectuer les opérations demandées. Dans le cas d'instructions conditionnelles, le code condition est renvoyé vers l'unité

microprogrammée afin de déterminer l'adresse de la micro-instruction suivante : mais puisque les unités de traitement sont indépendantes et que d'autre part les unités existent à plus d'un exemplaire, il est possible de déclencher une micro-instruction par cycle si cette micro-instruction s'adresse à des unités différentes par la fonction ou en nombre suffisant si elles sont de même type. Cependant, pour préserver la suite logique des opérations en cas d'anomalie, la séquence stricte est respectée par l'utilisation de tampons et de champs de contrôle.

Les microprogrammes sont réentrants et pour cela utilisent une pile contenant les adresses de déroutement de façon analogue à un programme.

CHAPITRE VII

UNITES PERIPHERIQUES ET DE CONTROLE

7.1 - UNITES PERIPHERIQUES

Les unités périphériques jouent vis à vis des unités de commande le rôle d'un superviseur, à la fois pour les commandes vers d'autres unités et vers les organes d'entrées-sorties. Leur nombre dépend de la puissance du système et varie de 2 à 16 pour les cas courants. Toutes ces unités sont identiques par leur conception de base mais il est possible de les distinguer par quelques mécanismes supplémentaires. Cependant, puisqu'aucune machine ne peut être unique (pour éviter les pannes), ces particularisations se trouvent sous forme de modules logiques transférables.

Le cycle d'une unité périphérique est de 50 ns, identique à celui des unités de commande.

7.1.1 - Composants d'une unité périphérique

7.1.1.1 - Mémoire

Chaque unité périphérique possède une mémoire à tores de 400 ns de temps de cycle entrelacée 4 fois, ce qui permet de transférer 2 mots de 32 bits en 2 cycles mineurs.

Chaque bloc comporte 8K octets, soit 32K octets tous adressables par unité.

Chaque mémoire est caractérisée par son numéro qui la rattache à l'unité périphérique correspondante.

7.1.1.2 - Unité de travail et adressage

L'unité de travail est semblable à celle d'un calculateur classique.

Les instructions sont microprogrammées dans une mémoire de 4 096 mots de 64 bits non destructible et une mémoire de 2 048 mots de 64 bits entièrement modifiable. La mémoire non destructible contient le jeu d'instructions standard du système. La seconde est réservée au superviseur.

L'adressage se fait par la méthode (base, déplacement), l'adressage est direct ou indirect.

Il existe 8 registres généraux pour toutes les opérations booléennes ou arithmétiques.

7.1.2 - Rôle d'une unité périphérique

L'unité périphérique, jouant le rôle d'un superviseur, a accès à tous les éléments du calculateur et peut modifier entre autres :

- les numéros de registres-pages en mémoire centrale,
- les numéros des registres-chapitres en mémoire étendue,
- le numéro d'affectation d'un banc en mémoire centrale,
- le mot d'état programme d'un calculateur.

Sur le plan local, elle peut charger la mémoire microprogrammée, commander les unités d'entrées-sorties.

Une unité périphérique peut en adresser une autre et modifier sa mémoire (concaténation de l'adresse et du numéro de l'unité).

Les demandes sont placées sur un système analogue à celui du "slot" du CDC 6600 toutes les 50 ns. Le temps d'accès étant alors doublé si la réponse n'est pas immédiatement satisfaite.

Les unités périphériques possèdent des interruptions hiérarchisées qui leur permettent de réagir aux signaux extérieurs.

Ces interruptions peuvent être armées ou désarmées par groupes pour les interruptions temps réel ou une par une pour les interruptions d'entrées-sorties.

Des signaux peuvent aussi être émis pour interroger d'autres unités périphériques qui répondent par interruption.

7.2 - UNITE DE LIAISON CANAL

L'unité de liaison canal assure la commande des canaux à partir des ordres transmis par les unités périphériques et contrôle l'information en provenance des unités d'entrées-sorties.

Nous avons indiqué que le nombre de canaux pouvait s'élever jusqu'à 64 pour les canaux multiplexés, 64 pour les canaux simples, 1 024 par groupes de 16 pour les canaux temps réel.

7.2.1 - Principe

Toute commande émise vers une unité d'entrée-sortie est prise en compte par l'unité de liaison qui déclenche le cycle d'opérations demandées. Lorsque celui-ci est terminé, l'unité de liaison envoie un signal d'interruption vers l'unité périphérique à l'origine de la demande.

Canaux multiplexés

Les canaux multiplexés permettent le fonctionnement simultané de plusieurs organes périphériques, chaque organe périphérique étant relié à un ou plusieurs sous-canaux. Tous les canaux sont bidirectionnels, et la quantité d'information transmise sur le sous-canal est l'octet.

Pour augmenter la vitesse du système, un même système peut être relié par plusieurs canaux, ce qui, en cas de charge excessive, permet de transférer une partie du travail sur l'autre. Ce transfert est effectué par l'unité de liaison.

Canaux simples

Ce sont des canaux rapides bidirectionnels qui ne permettent que l'utilisation d'un seul périphérique. Ils sont réservés par priorité aux organes à grand débit. Pour éviter le cas d'une panne, plusieurs canaux peuvent être reliés à une même unité : en cas de panne, les autres voies sont alors disponibles. Ces canaux sont flottants lorsqu'un groupe de canaux est affecté à un groupe d'organes périphériques. Dans ce cas, le choix est fait par l'unité de liaison.

Canaux temps réels

Ce sont des canaux bidirectionnels simples groupés 16 par 16. Ils permettent des transferts de données par interruption sur une voie :

à la réception d'un signal émis par un organe externe, l'unité de liaison envoie un code d'ordre indiquant le travail à effectuer.

7.2.2 - Composants de l'unité de liaison

L'unité de liaison est modulaire. Elle est entièrement micro-programmée et possède une mémoire pour emmagasiner les informations en provenance des unités périphériques.

Cette mémoire est constituée par des registres en circuits intégrés : lorsqu'une demande provient d'une unité périphérique, elle est enregistrée dans un registre ; cette demande étant mise sous une forme de micro-instruction, le décodage de cette micro-instruction provoque l'exécution du microprogramme associé.

7.3 - PROBLEME DE TRANSFERT

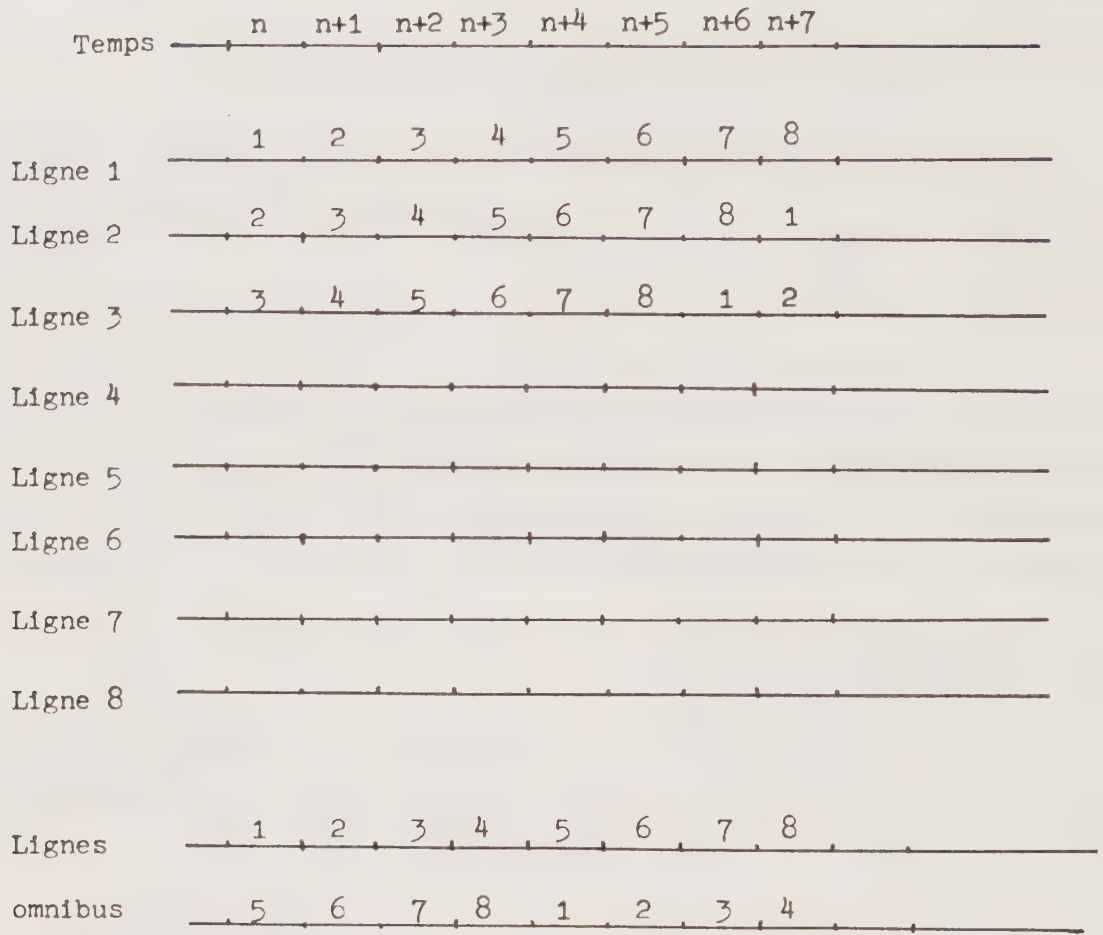
Les lignes de transferts de données ou de commande sont la partie la plus importante du système puisqu'elles sont à la base de son fonctionnement et de sa modularité (voir chapitre II).

Puisque les unités ne sont pas uniques, la défaillance de l'une ne peut entraîner l'arrêt du système : de même, il ne faut pas que la défaillance d'une ligne entraîne l'arrêt, d'où l'impératif de multiplier ces voies. La multiplication des voies entraîne alors le problème de la gestion et des procédures de reprises.

Le principe retenu est la codification des voies avec multiplexage dans le temps et dans l'espace.

7.3.1 - Transferts de données

Les voies de transferts de données (autres que celles d'entrées-sorties) ont un mot (64 bits + parité) et un octet de largeur.



A l'instant n , la voie 1 est affectée à l'unité de commande 1 et à l'instant $n+4$ à l'unité 5
Les deux lignes omnibus servent aux communications inter unités

Multiplexage spatio temporel

Fig 7-1

Le cycle des unités étant de 50 ns, vitesse maximum de transfert, entraîne l'obligation de lignes à 20 Mégabauds.

Cependant, l'utilisation d'une paire de lignes permet de réduire ce temps à 10 Mégabauds.

Le système de transfert emploie la technique pipeline avec indice (l'octet supplémentaire) afin de reconnaître les informations.

7.3.2 - Transferts de commandes

Les voies de transferts de données sont de types distincts suivant les éléments qu'elles relient. Cependant, elles utilisent la même technique, c'est-à-dire la fonction avec indice afin de reconnaître l'information.

Chaque unité de commande a sa propre ligne repérée par son numéro.

Il existe cependant d'autres lignes communes dont le mécanisme est le suivant :

Une adresse adressée par l'unité n à l'unité m sera placée dans un registre qui sera exploré de façon cyclique (400 ns) par un système comparable à celui du "tonneau". Cette même ligne sera interrogée par l'unité pour connaître les informations qui lui sont destinées.

Ce système fonctionnera en duplex avec le système de transfert de données communes. La panne de l'un ne bloquera pas le système.

7.3.3 - Problèmes de blocages, pannes

Afin d'éviter les blocages dus aux demandes simultanées, le système sera entièrement affecté à un moment donné (sauf pour les lignes communes). La reconnaissance se fera alors par l'intermédiaire du registre de liaison.

Il faut bien remarquer que certaines lignes de transfert peuvent se résumer à un registre frontière et n'impliquent pas forcément une ligne de câbles.

En cas de panne d'une ligne particulière, le système utilisera les autres lignes de la façon suivante : la demande sera distribuée de façon cyclique sur les lignes restantes et les modules affectés effectueront les mêmes recherches. Le cadencement sera le même, l'indice permettant de reconnaître la provenance de l'information (Technique du "vol" de cycle).

7.3.4 - Contrôle

Toutes les lignes de même nature sont regroupées en une même unité de contrôle qui assure le cadencement et le test des lignes. Cette unité est gérée par les calculateurs de contrôle.

7.4 - UNITES DE CONTROLE

Nous avons postulé que le système devait fonctionner sans arrêt même en cas de panne partielle. Pour cela, le contrôle des différentes unités composant ce système sera effectué par l'intermédiaire de deux calculateurs spécialisés, l'unité de contrôle central et l'unité de contrôle périphérique.

7.4.1 - Unité de contrôle central

L'unité de contrôle central a à sa charge la détection des pannes pouvant affecter le système. Pour cela, elle peut opérer de deux manières différentes :

- si la panne a été détectée sur une unité, celle-ci est provisoirement découplée et affectée à l'unité de contrôle qui, par le jeu de programmes, peut alors tester la circuiterie. Ces tests permettent de déterminer l'endroit précis de la panne qui est alors affichée sur l'écran de visualisation associé à l'unité de contrôle.
- si aucune panne n'existe, l'unité de contrôle utilise les temps morts du système pour tester tel ou tel équipement périphérique suivant un cahier de maintenance programmée.

Toute implantation nouvelle implique d'abord un test par l'unité de contrôle, c'est-à-dire que la mise en route d'une unité en cours de travail est possible puisqu'elle ne sera raccordée qu'avec l'autorisation de l'unité de contrôle.

L'unité de contrôle central est aussi chargée d'établir le cahier de bord et les statistiques relatives au blocage : chaque fois qu'un blocage intervient par surcharge d'un élément, un compteur caractéristique est incrémenté et, de cette façon, le système peut automatiquement signaler les goulots d'étranglement.

7.4.2 - Unité de contrôle périphérique

L'unité de contrôle périphérique joue vis à vis des unités périphériques et de l'unité de liaison un rôle identique à celui de l'unité de contrôle central.

De plus, elle sert à mettre le système en route par implantation des commandes convenables.

7.4.3 - Partage des contrôles et caractéristiques

Les unités de contrôle partagent la maintenance des voies d'accès du système entre leurs éléments propres. De plus, elles peuvent se tester l'une l'autre pour détecter leurs propres pannes.

Les unités de contrôle possèdent une mémoire de 32K octets partagée en deux : chaque unité de contrôle peut cependant accéder à l'autre mémoire. A chaque unité sont associés une console de visualisation, un système de disques contenant les programmes de tests et d'établissement du cahier de bord ainsi qu'une imprimante pour les états.

Dans une configuration réduite, il est possible au prix de quelques modifications de n'avoir qu'une seule unité de contrôle.

CHAPITRE VIII

SYSTEME DE COMMANDE ET PROGRAMMATION

8.1 - SYSTEME DE COMMANDE

Un point délicat est le système de commande.

En effet, pour que l'efficacité soit la plus grande possible, il faut qu'il se réduise au strict minimum afin de ne pas trop utiliser de temps machine.

Pour cela, les fonctions du système devront être bien déterminées car plusieurs programmes étant en exécution simultanément, les conflits d'accès au système devront être réglés sur une base de priorité.

Cependant, il est possible d'affecter un sous-système par unité périphérique, le sous-système étant plus particulièrement chargé de gérer une unité de commande. Alors la gestion du système reporte les conflits au niveau des sous-systèmes.

Il faut bien voir que le système de commande est réparti entre les différentes unités périphériques. La gestion des différentes tables correspond à un algorithme de files d'attente.

8.1.1 - Ordonnancement

Si les programmes n'ont aucun point commun, l'algorithme d'ordonnancement est relativement facile, chaque programme recevant un ordre de priorité.

Si les programmes ont des points communs, il est plus difficile d'imposer une priorité au risque de bloquer le système prioritaire en attente d'un système moins prioritaire.

Ce problème des "étreintes mortelles" est le plus difficile à résoudre : A a besoin d'une ressource utilisée par B, mais celui-ci a besoin du résultat de A pour continuer à travailler. Il faut donc que le système soit capable de lever ce genre d'ambiguïté en vérifiant l'inexistence de ce type de demandes croisées.

Temps d'exécution

Il est difficile de déterminer le temps d'exécution d'un programme, celui-ci dépendant du nombre d'itérations des boucles qui peut ne pas être connu par suite des critères de précision imposés, ou l'utilisation de fonctions internes dont la vitesse dépend à la fois des données et de la commande. De sorte que ce temps ne peut être qu'estimé.

Le deuxième point est que lors d'un défaut du système, le temps d'exécution peut être notablement allongé sans pour cela être prévisible.

Manipulation des signaux

Le système doit être capable de répondre aux signaux dont l'arrivée ne peut être prédite, ces signaux pouvant être hautement prioritaires (cas des unités tombant en panne).

Le problème est donc de savoir manipuler un ensemble de files d'attente dont les éléments ont des priorités pouvant varier et dont les instants d'arrivée dans la file sont imprévisibles.

8.1.2 - Problèmes liés aux programmes

Il existe plusieurs types de problèmes :

- aucune fonction de dégradation ou limite n'est associée aux tâches et dans ce cas, le seul problème est de garder le système le moins inactif possible.
- les fonctions de dégradation dépendant linéairement du temps existent parmi les tâches et il est nécessaire d'arranger les travaux par ordre décroissant d'urgence et de prédistriber les tâches de façon telle que la prochaine tâche soit attribuée au premier calculateur libre et garder la possibilité d'effectuer certaines permutations.
- les fonctions de dégradation ne dépendent pas linéairement du temps ; dans ce cas, le problème est plus compliqué et les priorités doivent être affectées suivant des critères à déterminer.
- une hiérarchie de priorités est imposée entre certaines tâches.
Dans ce cas, on les arrange par ordre de priorité et on les affecte à mesure que les calculateurs deviennent disponibles.
- une tâche nécessite tout ou partie de l'équipement et peut être partitionnée ; il peut y avoir avantage à les traiter simultanément dans le cas où il n'existe que des tâches moins prioritaires.

Cependant, il est aussi possible de considérer le fait que certains travaux demandent beaucoup de calculs, peu d'entrées-sorties et d'autres beaucoup d'entrées-sorties et peu de calculs ; leur affectation simultanée permet alors de remplir les conditions optimales.

8.1.3 - Problèmes liés à l'environnement

Plusieurs cas peuvent se présenter :

- les travaux, compte tenu des limites vues précédemment, sont connus par avance,
- les travaux ne sont pas connus, mais on connaît à peu près leur nombre et l'instant d'arrivée, ainsi que leur nature et leur temps d'exécution,
- il y a des problèmes liés aux facteurs humains (montage de bandes, alimentation en cartes, papier pour imprimantes).

Dans le premier cas, il est possible d'établir un algorithme permettant une optimisation du travail à effectuer, en tenant compte des relations de précédences pour minimiser les pertes de temps.

Dans le deuxième cas, on réordonne périodiquement les travaux en fonction des renseignements parvenus au fur et à mesure de l'arrivée des tâches.

Dans le troisième cas, le système a des travaux arrivant avec des priorités variables d'une manière relativement aléatoire. Le système doit alors être capable de répondre instantanément de la façon la plus efficace.

8.1.4 - Moniteur d'exécution et segmentation

Le moniteur d'exécution est la seule partie résidant de façon permanente en mémoire. Les problèmes implantés ne nécessitent pas d'être complets et peuvent être partitionnés. Cependant, chaque segment doit pouvoir être exécuté indépendamment une fois que les données lui ont été fournies.

La taille d'un programme est un problème critique car plus le nombre de segments est grand, plus de temps est perdu par le moniteur dans leur manipulation.

A la fin d'un travail, le moniteur réaffecte les disponibilités à la tâche la plus prioritaire et provoque le chargement des modules à partir des fichiers du système. Si le programme ne tient pas en mémoire, il peut lui affecter de la mémoire disponible ou charger un autre programme. L'utilisation d'une mémoire étendue permet de faciliter ce travail, par la possibilité de préparer par avance les différentes tâches.

Le moniteur répond aussi aux interruptions en provenance des unités de commande pour effectuer les entrées-sorties ou les commutations nécessaires.

La comptabilité du système est aussi une part importante du travail du moniteur, la difficulté étant liée à la segmentation des disponibilités et à la difficulté de pénaliser quelqu'un par suite de l'indisponibilité d'une partie du système.

8.2 - PROGRAMMATION

En dehors du jeu d'instructions arithmétiques traditionnel, il existe un certain nombre d'instructions spécifiques de multi-traitement. Leur détermination conditionne les performances du système ; elles doivent tendre à faciliter les communications entre éléments :

- liaison superviseur - unité de commande,
- liaison unité de commande à unité de commande,
- liaison superviseur - unité de liaison.

Les instructions du superviseur sont celles des unités périphériques, celles des unités de commande étant des appels au superviseur.

Instructions superviseur

Le jeu d'instructions des unités périphériques en dehors des instructions courantes contient un certain nombre de fonctions spécialisées :

- modification des numéros des registres mémoire,
- modification des affectations de modules mémoire,
- chargement des modules mémoire,
- chargement du mot d'état programme et des registres pour une unité de commande,
- modification du mot d'état programme,
- commande des organes d'entrée-sortie par l'unité de liaison.

La commande des entrées-sorties est facilitée par l'utilisation de fonctions microprogrammées dans l'unité de liaison dont les arguments lui sont fournis par l'unité périphérique.

Instructions unités de commande

Ce sont surtout, en dehors des instructions arithmétiques :

- l'appel du superviseur,
- les fonctions microprogrammes,
- les interruptions internes programmées.

8.2.1 - Compilation et assemblage

Assemblage

L'assemblage des instructions se fait de manière traditionnelle ; cependant, la possibilité d'utiliser des micro-instructions introduit une plus grande latitude et une plus grande souplesse.

Une phase complète d'assemblage comporte :

- la définition des microprogrammes (pseudo instruction MICRO),
- la définition des instructions (pseudo instruction MACRO).

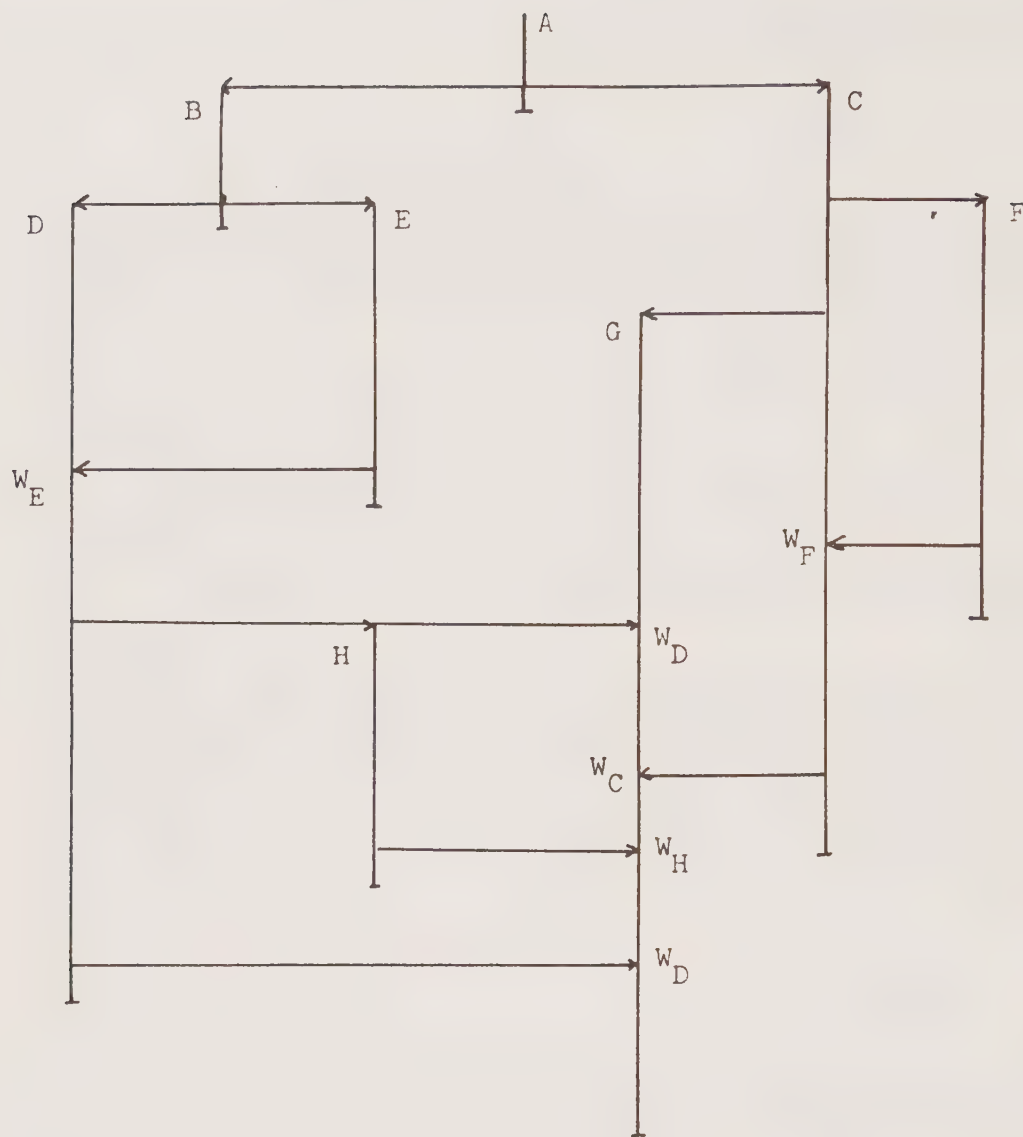
La possibilité d'intégrer des micro-instructions en cours de programme fait que le micro-assembleur est une partie intégrale du macro-assembleur puisque tous les microprogrammes doivent être écrits de façon réentrante ce qui permet à l'intérieur d'un micro-programme d'en appeler un autre.

Compilation

Ce calculateur étant un calculateur universel, tous les compilateurs sont possibles : certains peuvent intégrer des microprogrammes pour faciliter l'analyse des chaînes et les langages de listes sont faciles à manipuler puisque leurs instructions peuvent être microprogrammées.

8.2.2 - Simulation et émulation

Par l'existence des microprogrammes, la simulation et l'émulation de machines ou de mécanismes particuliers sont aisées.



→ Initialisation d'une chaîne ou transmission d'information

⊥ Fin de chaîne

W_A Attente transmission chaîne A

Exemple d'exécution parallèle

Fig 8-1

La simulation ou l'émulation des entrées-sorties qui sont difficiles habituellement sont ici facilitées par la possibilité de déclencher des interruptions internes par les ordres spéciaux d'interruption programmée.

8.2.3 - Assemblage pour les unités périphériques

Il existe pour les unités périphériques un micro et un macro-assembleur de la même manière que pour les unités de commande.

Ces programmes permettent l'assemblage des programmes destinés aux unités périphériques et aux calculateurs de contrôle. En effet, les instructions des calculateurs de contrôle ont beaucoup de points communs avec celles des unités périphériques et utilisent le même langage.

Il existe aussi des langages spécialisés pour la transmission, le télétraitement, le temps partagé qui sont compilés au niveau des unités périphériques.

CHAPITRE IX

CONCLUSION

Dans ce travail, nous avons donné le principe de la construction d'un système complexe pouvant, sans difficulté, s'adapter aux besoins des utilisateurs par adjonction d'éléments de structure plus ou moins élaborée. Ce système complexe utilise des mécanismes relativement simples et parfois connus. Cette simplicité des mécanismes apporte une grande souplesse dans la construction et une facilité de connexion obtenue grâce à l'emploi d'éléments de structure adaptée. Souplesse et facilité de connexion permettent en particulier la résolution aisée des problèmes provoqués par les perturbations créées par les pannes et blocages. Les éléments constitutants sont conçus indifféremment comme des circuits ou des programmes, cette dualité étant maintenue à tous les niveaux. Cette particularité permet en cas de défection d'un élément de le remplacer soit par un circuit équivalent, soit par un programme sans que cela affecte le fonctionnement du système tout entier autrement que par un accroissement du temps de traitement.

L'organisation des éléments composants le système représente dans notre travail la principale originalité. Elle pose des problèmes combinatoires de logique booléenne, de structures de files d'attente. La nature des problèmes de logique est inséparable de la conception générale de la circuiterie et de la génération des signaux de commande. Les structures de files d'attente sont liées à l'existence des différentes unités de traitement et aux demandes émises par les différents

programmes travaillant en concurrence. L'organisation du système de commande tient compte de ces différents éléments pour la répartition des tâches et la possibilité de manipuler de vastes fichiers sans perte de temps ou d'information pour la plus grande satisfaction des besoins, un coût moins élevé et une sûreté plus grande qu'avec des systèmes traditionnels.

BIBLIOGRAPHIE

I - CLASSEMENT ALPHABETIQUE

1 - ABERTH O.

A multiple computer linkage

IEEE Tr. - vol. C 18 n° 12 - Dec. 1969 - pp. 1132 - 1134

2 - ANDERSON J.P.

Program structure for parallel processing

Comm. ACM - vol. 8 - Dec. 1965 - pp. 786 - 788

3 - ANDERSON D.W., SPARACIO F.J., TOMASULO R.M.

The IBM System/360 Model 91 : Machine Philosophy and
Instruction Handling

IBM R. & D. - vol. 11 n° 1 - Jan. 1967 - pp. 3 - 24

4 - ANDERSON S.F., EARLE J.G., GOLDSCHMIDT R.E., POWERS D.M.

The IBM System/360 Model 91 : Floating point execution unit

IBM R. & D. - vol. 11 n° 1 - Jan. 1967 - pp. 34 - 53

5 - ARDEN B.W., GALLER B.A., O'BRIEN T.C., WESTERVELT F.H.

Program and addressing structure in a time sharing environment

J. ACM - vol. 13 n° 1 - Jan. 1966 - pp. 1 - 16

6 - ARSAC J.

Les systèmes de conduite des ordinateurs

DUNOD - 1968

7 - ASCHENBRENNER R.A., FLYNN M.J., ROBINSON G.A.

Intrinsic multiprocessing

AFIPS SJCC - vol. 30 - 1967 - pp. 81 - 86

- 8 - ASCHER J.R., SKINNER C.E.
Effects of storage contention on system performance
IBM Sys. J. n° 4 - 1969 - pp. 319 - 333
- 9 - BAER J.L., BOVETT D.P.
Compilation of arithmetic expression for parallel computations
IFIP - 1968 - Software I Booklet B - pp. 4 - 10
- 10 - BALDWIN F.R., GIBSON W.B., POLAND C.B.
A multiprocessing approach to a large computer system
IBM Sys. J. - vol. I - Sept. 1962
- 11 - BARNES G.H., BROWN R.M., KATO M., KUCK D.J., SLOTNICK D.L.,
STOKES R.A.
The ILLIAC IV Computer
IEEE Tr. - vol. C 17 n° 8 - Aug. 1968 - pp. 746 - 757
- 12 - BAUER W.F.
Why multi-computers ?
Datamation - vol. 8 n° 9 - Sept. 1962 - pp. 51 - 55
- 13 - BELADY L.A.
A study of replacement algorithms for a virtual storage computer
IBM Sys. J. - vol. 5 n° 2 - 1966 - pp. 78 - 101
- 14 - BERNSTEIN A.J.
Analysis of programs for parallel processing
IEEE Tr.-vol. EC 15 n° 5 - Oct. 1966 - pp. 757 - 763
- 15 - BEUSCHER H.J., TOY W.N.
Check Schemes for Integrated Microprogrammed Control and Data
Transfer Circuitry
IEEE Tr. - vol. C 19 n° 12 - Dec. 1970 - pp. 1153 - 1159

- 16 - BIBB J., ESTRIN G., RUSSEL B., TURN R.
Parallel processing in a restructurable computer system
IEEE Tr. - vol. EC 12 n° 5 - 1963 - pp. 747 - 754

- 17 - BIBB J., ESTRIN G., RUSSEL B.
The UCLA variable structure computer system
Proc. of 1962 Workshop on Computer Organisation
Ed. A. Barnum & M. Knapp - Spartan Books - 1963 - pp. 182 - 190

- 18 - BOLAND L.J., GRANITO G.D., MARCOTTE A.U., MESSINA B.U.,
SMITH J.W.
The IBM System/360 Model 91 : Storage system
IBM R. & D. - vol. II n° 1 - Jan. 1967 - pp. 54 - 68

- 19 - BORCK W.C., GREGORY J.G., HUDSON J.R., MURTHA J.C.
The design of a variable structure parallel network computer
Prospects for simulation and simulators of dynamic systems -
pp. 227 - 250

- 20 - BORCK W.C., Mc REYNOLDS R.C., SLOTNICK D.L.
The Solomon Computer : a preliminary report
Proc. of 1962 Workshop on Computer Organisation
Ed. A. Barnum & M. Knapp - Spartan Books - 1963 - pp. 66 - 92

- 21 - BRAWN B., GUSTAVSON F.
Program behavior in a paging environment
AFIPS FJCC - 1968 - vol. 33 - pp. 1019 - 1032

- 22 - BROWN J.L., DOODY D.T., FUGG P.
IBM System/360 Engineering
AFIPS SJCC - 1964 - vol. 26 - p. 205

23 - BRYAN G.I.

Joss : 20 000 hours at a console - a statistic summary
AFIPS FJCC - 1967 - vol. 31 - pp. 769 - 777

24 - BURNETT G.J., WYLES H.

Some relationship between failure detection probability
and computer system reliability
AFIPS FJCC - 1967 - vol. 31 - pp. 745 - 756

25 - BURNETT G.J., HOKOM R.A., KOCZELA L.J.

A distributed system for general purpose computing
AFIPS FJCC - 1967 - vol. 31 - pp. 757 - 768

26 - CAMPEAU J.O.

Block oriented computer
IEEE Tr. - vol. C 18 - n° 8 - Aug. 1969 - pp. 706 - 718

27 - CANTRELL E.

Multiprogramming system performance measurement and evaluation
AFIPS FJCC - 1968 - vol. 32

28 - CHANG H.Y., SCANLON J.M.

Design principles for processor maintainability in real time
systems
AFIPS FJCC - 1969 - vol. 35 - pp. 319 - 328

29 - CHANG W.A.

A queuing model for a simple case of time-sharing
IBM Sys. J. - vol. 5 n° 2 - 1966 - pp. 115 - 125

30 - CHEN E.C.Y., HALEY G., SUMNER F.H.

The central control unit of the Atlas computer
IFIP - 1962 - North Holland - pp. 657 - 663

31 - CHIN TUNG

On the apparent continuity of processing in a paging environment
IEEE Tr. - vol. C I9 n° II - Nov. I970 - pp. I047 - I054

32 - CHU W.W.

Optimal file allocation in a multiple computer system
IEEE Tr. - vol. C I8 n° IO - Oct. I969 - pp. 885 - 889

33 - CODD E.F.

Multiprogramming
Advances in computers - vol. 3 - I962 - Ed. F. Alt.
Academic Press

34 - COFFMAN E.G., SCHWARTZ J.I., WEISMAN C.

A general purpose time sharing computer
ACM 22nd Nat. Conf. - I967 - pp. 263 - 272

35 - COFFMANN E.G.

Analysis of two time sharing algorithms designed for limited
swapping
J. ACM - vol. I5 n° 3 - July I968 - pp. 34I - 353

36 - COFFMANN E.G., VARIAN L.C.

Further experimental data on the behavior of programs in a
paging environment
Comm. ACM - vol. II n° 7 - July I968 - pp. 47I - 474

37 - COMEAU L.W., LINDQUIST A.B., SEEGER R.R.

A time sharing system using an associative memory
Proc. IEEE 54 n° I2 - Dec. I966 - pp. I774 - I779

38 - COMFORT W.T.

Highly parallel machines
Proc. of I962 Workshop in Computer Organisation
Ed. A. Barnum & M. Knapp - Spartan Books - I963 - pp. I26 - I55

- 39 - COMFORT W.T.
A computing system design for user service
AFIPS FJCC - 1965 - Vol. 27 - pp. 619 - 626

- 40 - CONTI C.J., GIBSON D.H., PITKOWSKY S.H.
Structural aspects of the System/360 model 85 : I General
Organisation
IBM Sys. J. - Vol. 7 n° 1 - 1968 - pp. 2 - 14

- 41 - CONWAY M.E.
A multi-processor system design
AFIPS FJCC - 1963 - Vol. 24 - pp. 139 - 146

- 42 - CORBATO F.J., GRAHAM R.M., VYSSOTSKY V.A.
Structure of the Multics supervisor
AFIPS FJCC - 1965 - Vol. 27 - Part I - pp. 203 - 212

- 43 - COTTON L.W.
Maximum rate pipeline system
AFIPS SJCC - 1969 - Vol. 34

- 44 - COTTON L.W.
Circuit implementation of high speed pipeline system
AFIPS FJCC - 1965 - Vol. 27 - pp. 489 - 504

- 45 - COULEUR J.F., GLASER E.L., OLIVER G.A.
System design of a computer for time sharing application
AFIPS FJCC - 1965 - Vol. 27 - Part I - pp. 197 - 202

- 46 - CRITCHLOW A.J.
Generalized multiprocessing and multiprogramming system
AFIPS FJCC - 1963 - Vol. 23 - pp. 107 - 126

47 - CURTIN W.A.

Multiple computer systems

Advances in Computers - Ed. F. Alt & M. Rubinoﬀ - vol. 4
pp. 245 - 303

48 - DAHM D.M., GERBSTADT F.H., PACELLI M.M.

A system organisation for resource allocation

Comm. ACM - vol. 10 n° 12 - Dec. 1967 - pp. 772 - 779

49 - DALEY R.C., DENNIS J.B.

Virtual memory, processes and sharing in Multics

Comm. ACM - vol. 11 n° 5 - May 1968 - pp. 306 - 312

50 - DEBRAINE P.

Adressage symbolique absolu - Structure facilitant le travail
en partage de temps

Rapp. CEA-R-3588 - 1968 - Doc. Française

51 - DELLER G.T.

A use of macros in translation of symbolic assembly language
of one computer to another

Comm. ACM - vol. 8 n° 12 - 1965

52 - DENNING P.J.

Effects of scheduling on file memory operations

AFIPS SJCC - 1967 - vol. 30 - pp. 9 - 22

53 - DENNING P.J.

Virtual memory

Comp. Sur. - 1970 - vol. 2 n° 3 - pp. 153 - 189

54 - DENNIS J.B.

Segmentation and the design of multiprogrammed computer systems

J. ACM - vol. 12 n° 4 - Oct. 1965 - pp. 589 - 602

- 55 - DENNIS J.B., VAN HORN E.C.
Programming semantics for multiprogrammed computations
Comm. ACM - vol. 9 n° 3 - March 1966 - pp. 143 - 155
- 56 - DIJKSTRA E.W.
The structure of the THE multiprogramming system
Comm. ACM - vol. 11 n° 5 - May 1968 - pp. 341 - 346
- 57 - DRUMMOND M.E.
A perspective on system performance evaluation
IBM Sys. J. - vol. 8 - 1969 - pp. 252 - 263
- 58 - EDWARDS D.G., KILBURN T., LANIGAN M.J., SUMNER F.H.
One level storage system
IEEE Tr. - vol. EC II n° 2 - 1962 - pp. 223 - 235
- 59 - ELROD T.H.
The CDC 7600 and Scope 76
Datamation - April 1970 - vol. 16 n° 4 - pp. 80 - 85
- 60 - ENGLAND H.W., MENDELSON M.J.
The SDS Sigma 7 : a real time sharing computer
AFIPS FJCC - 1966 - vol. 29 - pp. 51 - 64
- 61 - ERICKSON G.A., MUTSCHLER E.G., PICKERING G.E.
Multi-computer programming for a large scale real time data
processing system
AFIPS SJCC - 1964 - vol. 25 - pp. 445 - 462
- 62 - FINE G.H., JACKSON C.W., Mc ISAAC P.V.
Dynamic program behavior under paging
ACM 21 st Nat. Conf. - pp. 223 - 228

- 63 - FLORES I.
Virtual memories and paging
Datamation - Part I - vol. 13 n° 8 - Aug. 1967 - pp. 31 - 34
Part 2 - vol. 13 n° 9 - Sep. 1967 - pp. 41 - 48
- 64 - FLYNN M.J., LOW P.R.
The IBM System/360 model 91 : Some remarks on system development
IBM R. & D. - Vol. II n° 1 - Jan. 1967 - pp. 2 - 7
- 65 - FLYNN M.J., TJADEN G.J.
Detection and parallel execution of independant instructions
IEEE Tr. - Vol. C 19 n° 10 - pp. 889 - 895
- 66 - FLYNN M.J., TUCKER A.B.
Dynamic microprogramming : Processor organisation and programming
Comm. ACM - Vol. 14 n° 4 - April 1971 - pp. 240 - 250
- 67 - FOSTER C.C.
Determination of priority in associative memory
IEEE Tr. - Vol. C 17 n° 8 - Aug. 1968 - pp. 788 - 789
- 68 - FOSTER C.C.
Uncoupling central processor and storage device speeds
Computer J. - Vol. 14 n° 1 - pp. 45 - 48
- 69 - FOTHERINGHAM J.
Dynamic storage allocation in the ATLAS computer including an automatic use of a backing store
Comm. ACM - Vol. 4 n° 10 - Oct. 1961 - pp. 435 - 436
- 70 - FRIEDMAN T.D., YANG S.C.
Methods used in an automatic logic design generator (ALERT)
IEEE Tr. - Vol. C 18 n° 7 - July 1969 - pp. 593 - 614

- 71 - FUCHEL K., HELLER S.
Considerations in the design of a multiple computer system
with ECS
Comm. ACM - vol. II n° 5 - May 1968 - pp. 334 - 340
- 72 - FULLER R.H.
Associative parallel processing
AFIPS SJCC - 1967 - vol. 30 - pp. 471 - 476
- 73 - GARNER H.L., SQUIRE J.S.
Iterative circuit computer
Proc. of 1962 Workshop on Computer Organisation
Ed. A. Barnum & M. Knapp - Spartan Books - pp. 156 - 181
- 74 - GECSEI J., MATTSON R.L., SLUTZ D.R., TRAIGER I.L.
Evaluation techniques for storage hierarchies
IBM Sys. J. - vol. 9 n° 2 - 1970 - pp. 78 - 117
- 75 - GIBSON C.T.
Time sharing in the IBM System/360 : Model 67
AFIPS SJCC - 1966 - vol. 28 - pp. 61 - 78
- 76 - GIBSON D.H.
Considerations in block oriented system design
AFIPS SJCC - 1967 - vol. 30 - pp. 75 - 80
- 77 - GRAHAM R.M.
Protection in an information processing utility
Comm. ACM - vol. II n° 5 - May 1968 - pp. 365 - 369
- 78 - GRAHAM W.R.
The parallel and pipeline computer
Datamation - April 1970 - vol. 16 n° 4 - pp. 68 - 71

79 - GREEN J.

Microprogramming, emulators and programming languages
Comm. ACM - vol. 9 n° 3 - 1966 - pp. 230 - 232

80 - GUPTA S.C., SHEMER J.E.

A simplified analysis of processor look ahead and simultaneous
operator of a multi-module mass memory
IEEE Tr. - vol. C 18 n° 1 - Jan. 1969 - pp. 64 - 71

81 - GUPTA S.C., SHEMER J.E.

On the design of bayesian storage allocation algorithms for
paging and segmentation
IEEE Tr. - vol. C 18 n° 7 - July 1969 - pp. 644 - 651

82 - HABERMAN A.N.

Prevention of system deadlocks
Comm. ACM - vol. 12 n° 7 - July 1969 - pp. 373 - 377

83 - HANLON A.G.

Content adressable and associative memory systems : a survey
IEEE Tr. - vol. EC 15 n° 4 - Aug. 1966 - pp. 509 - 521

84 - HEBENSTREIT J.

Générateur de microprogrammes
Thèse Doct. ès Sciences - Paris - Mai 1969

85 - HELLERMAN L.

Some principles of time sharing scheduler strategies
IBM Sys. J. - vol. 8 n° 2 - 1969 - pp. 94 - 117

86 - HELLERMAN L.

Parallel processing of algebraic expressions
IEEE Tr. - vol. EC 15 - Feb. 1966

- 87 - HELLERMAN L., HOERNES G.E.
Control storage use in implementing an associative memory
for a time shared processor
IEEE Tr. - vol. C I7 n° I2 - Dec. 1968 - pp. II44 - II5I
- 88 - HENNIE F.C.
Finite state models for logical machines
J. Wiley - N.Y. - 1968
- 89 - HOLLANDER G.L.
Architecture for large computer systems
AFIPS SJCC - 1967 - vol. 30 - pp. 463 - 466
- 90 - HOLT A.W.
Program organisation and record keeping for dynamic storage
allocation
Comm. ACM - vol. 4 n° IO - Oct. 196I - pp. 422 - 43I
- 91 - HORWITZ L.P., KARP R.M., MILLER R.E., WINOGRAD S.
Index register allocation
ACM J. - vol. I3 n° I - Jan. 1966 - pp. 43 - 6I
- 92 - HOWARTH D.J., JONES P.D., WYLD M.T.
The ATLAS scheduling system
AFIPS SJCC - 1963 - vol. 23 - pp. 59 - 67
- 93 - HUMPHREY T.A.
Large core storage utilisation in theory and practice
AFIPS SJCC - 1967 - vol. 30 - pp. 7I9 - 727
- 94 - HUSSON S.S.
Microprogramming : principles and practices
Prentice Hall - 1970

- 95 - ILIFFE J.K., JODEIT J.G.
A dynamic storage allocation scheme
Computer J. - vol. 5 n° 3 - Oct. 1962 - pp. 200 - 209
- 96 - IRONS E.T.
A rapid turnaround multiprogramming system
Comm. ACM - vol. 8 n° 3 - March 1965 - pp. 152 - 157
- 97 - IVERSON K.E.
A programming language
J. Wiley - N.Y. - 1962
- 98 - JOHNSON R.R.
Measures and evaluation
1969 - NATO - Summer School
- 99 - KEEFE D.D.
Hierarchical control programs for systems evaluation
IBM Sys. J. - vol. 7 n° 2 - 1968 - pp. 123 - 133
- 100 - KINNIMENT D.J., KNOWLES H.E., LAVINGTON S.H.
An experimental paging unit
Computer J. - vol. 14 n° 1 - pp. 55 - 60
- 101 - KLEINROCK L.
Swap time considerations in time shared systems
IEEE Tr. - vol. C 19 n° 6 - pp. 534 - 540
- 102 - KOCZELA L.J., WANG G.Y.
The design of a highly parallel computer organisation
IEEE Tr. - vol. C 18 n° 6 - June 1969 - pp. 520 - 529

- 103 - KOCZELA L.J.
The distributed processor organisation
Advances in Computers - Ed. F. Alt & M. Rubinoff - vol. 9
pp. 286 - 360
- 104 - KOHAVI Z., LAVALLEE P.
Design of diagnosable sequential machines
AFIPS SJCC - 1967 - vol. 30 - p. 713
- 105 - KUCK D.J.
A preprocessing high speed memory system
IEEE Tr. - vol. C 19 n° 9 - pp. 793 - 801
- 106 - KUEHNER C.J., RANDELL B.
Dynamic storage allocation systems
Comm. ACM - vol. 11 n° 5 - May 1968 - pp. 297 - 306
- 107 - KUEHNER C.J., RANDELL B.
Demand paging in perspective
AFIPS FJCC - 1968 - vol. 33 - pp. 1011 - 1018
- 108 - LAMPSON B.W.
A scheduling philosophy for multiprocessing systems
Comm. ACM - vol. 11 n° 5 - May 1968 - pp. 347 - 360
- 109 - LASSER D.J.
Productivity of multiprogrammed computers - Progress in
developing an analytic prediction method
Comm. ACM - vol. 12 n° 12 - Dec. 1969 - pp. 678 - 684
- 110 - LAVER H.C.
Bulk core in the 360/67 time sharing system
AFIPS SJCC - 1967 - vol. 30 - pp. 29 - 40

- 111 - LAWSON H.W.
Programming language oriented instructions streams
IEEE Tr. - vol. C I7 - I968 - pp. 476 - 485
- 112 - LEE F.S., Mc NAUGHTON P.C.
The virtual multiprocessor - a new computer architecture
IFIP - I968 - Booklet E - pp. 39 - 4I
- 113 - LEHMAN M.M.
A survey of problems and preliminary results concerning
parallel processing and parallel processors
IEEE - vol. 54 n° I2 - Dec. I966 - pp. I889 - I90I
- 114 - LEHMAN M.M., ROSENFELD J.L.
Performance of a simulated multiprogramming system
AFIPS FJCC - I968 - vol. 33 - pp. I4I - I42
- 115 - LEHMAN M.M., SHEDLER G.S.
Evaluation of redundancy in a parallel algorithm
IBM Sys. J. - vol. 6 n° 3 - I967 - pp. I42 - I49
- 116 - LINDQUIST A.B., SEEGER R.R.
Associative logic for highly parallel systems
AFIPS SJCC - I964 - vol. 24 - pp. 489 - 493
- 117 - LIPTAY J.S.
Structural aspects of the system/360 model 85 : II The Cache
IBM Sys. J. - vol. 7 n° I - I968 - pp. I5 - 2I
- 118 - LITWIN S., PRYWES N.S.
The multi-list central processor
Proc. of the I962 Workshop on Computer Organisation
Ed. A. Barnum & M. Knapp - Spartan Books - I963 - pp. 2I4 - 242

119 - MADNICK S.E.

Multiprocessor software lockout

ACM 23rd Nat. Conf. - pp. 19 - 24

120 - MAHER R.S.

Problems of storage allocation in multiprocessor multiprogrammed system

Comm. ACM - vol. 4 n° 10 - Oct. 1961 - pp. 421 - 422

121 - MANNING E.

On computer self diagnosis :

Part I - Experimental study of processor

Part II - Generalisation and design principles

IEEE Tr. - vol. EC 15 n° 6 - Dec. 1966

122 - Mc CULLOUGH J.D., SPEIERMAN K.H., ZURCHER F.W.

A design for a multiple user multiprocessing system

AFIPS FJCC - 1965 - vol. 27 - Part I - pp. 611 - 617

123 - Mc DOUGALL M.H.

Simulation of an ECS based operating system

AFIPS SJCC - 1967 - vol. 30 - p. 735

124 - Mc GEE W.C.

On dynamic program relocation

IBM Sys. J. - vol. 4 n° 3 - 1965 - pp. 184 - 199

125 - MEADE R.M.

On memory system design

AFIPS FJCC - vol. 37 - 1970 - pp. 33 - 43

126 - MINSKY M.L.

Computation : Fine and infinite machines

Prentice Hall - 1967

- 127 - MURPHEY J.O., WADE R.M.
The IBM 360/I95
Datamation - vol. 16 n° 4 - April 1970 - pp. 72 - 79
- 128 - MURTHA J.C.
Highly parallel information processing systems
Advances in Computers - Ed. F. Alt & M. Rubinoﬀ
vol. 7 - pp. 2 - II6
- 129 - NATARAJAN N.K., THOMAS P.A.V.
A multiaccess associative memory
IEEE Tr. - vol. C 18 n° 5 - May 1969 - pp. 424 - 428
- 130 - O'NEILL R.W.
Experience using a time shared multiprogramming system with
dynamic address relocation hardware
AFIPS SJCC - 1967 - vol. 30 - pp. 6II - 622
- 131 - PADEGS A.
Structural aspects of the system/360 model 85 : III Extensions
to floating point architecture
IBM Sys. J. - vol. 7 n° I - 1968 - pp. 22 - 29
- 132 - PYKE T.N.
Time shared computer systems
Advances in Computers - Ed. F. Alt & M. Rubinoﬀ - vol. 8
- 133 - RANDELL B.
A note on storage fragmentation and program segmentation
Comm. ACM - vol. 12 n° 7 - July 1969 - pp. 365 - 369
- 134 - RAMAMOORTHY C.V.
A structural theory of machine diagnosis
AFIPS SJCC - 1967 - vol. 30 - p. 743

- 135 - RAMAMOORTHY C.V.
The analytic design of a dynamic look ahead and program
segmenting for multiprogrammed computers
ACM 21st Nat. Conf. - 1966 - pp. 229 - 239
- 136 - RAMAMOORTHY C.V., M. TSYCHIYA
A study of user microprogrammable computer
AFIPS SJCC - 1970 - vol. 36 - pp. 165 - 181
- 137 - REITER R.
Scheduling parallel computations
J. ACM - vol. 15 n° 4 - Oct. 1968
- 138 - RODER J., ROSENE A.F.
Memory protection in multiprocessing systems
IEEE Tr. - vol. EC 16 n° 3 - June 1967 - pp. 320 - 326
- 139 - ROSENE A.F.
Memory allocation for multiprocessors
IEEE Tr. - vol. EC 16 n° 5 - Oct. 1967 - pp. 659 - 665
- 140 - ROSENFELD J.L.
A case study in programming for parallel processors
Comm. ACM - vol. 12 n° 12 - Dec. 1969 - pp. 645 - 655
- 141 - ROSIN R.F.
Supervisory and monitor systems
Comp. Sur. - 1969 - vol. 1 n° 1 - pp. 37 - 54
- 142 - ROSIN R.F.
Contemporary concept of microprogramming and emulation
Comp. Sur. - 1969 - vol. 1 n° 4 - pp. 197 - 212

143 - SAYRE D.

Is automatic folding of program efficient enough to displace manual ?

Comm. ACM - vol. 12 n° 12 - Dec. 1969 - pp. 656 - 660

144 - SCHLAEPPI H.P.

A formal language for describing machine logic timing and sequencing (LOTIS)

IEEE Tr. - vol. EC 13 - Aug. 1964 - pp. 439 - 448

145 - SCHON H.

Computer aided digital system design and analysis using a register transfer language

IEEE Tr. - vol. EC 13 - Dec. 1964 - pp. 730 - 737

146 - SCHWARTZ J.

Large parallel computers

J. ACM - vol. 13 n° 1 - Jan. 1966 - pp. 23 - 32

147 - SEAMAN P.H., SOUCY R.C.

Simulating operating systems

IBM Sys. J. - vol. 8 n° 4 - 1969 - pp. 264 - 279

148 - SENZIG D.N.

Observations on high performance machine

AFIPS FJCC - 1967 - vol. 31 - pp. 791 - 799

149 - SHEMER J.E., SHIPPEY G.A.

Statistical analysis of paged and segmented computer systems

IEEE Tr. - vol. EC 15 n° 6 - Dec. 1966 - pp. 855 - 863

150 - SHERLOCK J.F.

The simulation of a multi-computer system

IEEE Tr. - vol. C 19 n° 11 - Nov. 1970 - pp. 1114 -

- 151 - SMITH J.L.
Multiprogramming under a page on demand access
Comm. ACM - vol. 10 n° 10 - Oct. 1967 - pp. 636 - 646
- 152 - STABLER E.P.
Microprogram transformations
IEEE Tr. - Vol. C 19 n° 10 - Oct. 1970 - pp. 908 - 916
- 153 - STANGA D.C.
UNIVAC II08 multiprocessor system
AFIPS SJCC - 1967 - vol. 30 - pp. 67 - 74
- 154 - STEVENS D.F.
On overcoming high priority paralysis in multiprogramming
systems : a case history
Comm. ACM - vol. 11 n° 8 - Aug. 1968 - pp. 539 - 541
- 155 - STONE H.S.
The organisation of high speed memory for parallel block
transfer of data
IEEE Tr. - vol. C 19 n° 1 - Jan. 1970 - pp. 47 - 53
- 156 - STONE H.S.
A logic-in-memory computer
IEEE Tr. - vol. C 19 n° 1 - Jan. 1970 - pp. 73 - 78
- 157 - THORLIN J.F.
Code-generation for parallel instruction execution computers
AFIPS SJCC - 1967 - vol. 30 - p. 641
- 158 - THORNTON J.E.
Parallel operation in the CDC 6600
AFIPS SJCC - 1964 - vol. 26 - Part 2 - p. 33

159 - TOMASULO A.M.

An efficient algorithm for exploiting multiple arithmetic units

IBM R. & D. - vol. II n° I - Jan. 1967 - pp. 25 - 33

160 - TSUJIGADO M.

Multiprogramming, swapping and program residence priority in the FACOM 230-60

AFIPS SJCC - 1968 - vol. 32 - pp. 223 - 228

161 - TUCKER S.G.

Emulation of large systems

Comm. ACM - vol. 8 n° 12 - 1965 - pp. 753 - 761

162 - TUCKER S.G.

Microprogram control for system/360

IBM Sys. J. - 1967 - vol. 6 n° 4 - pp. 222 - 241

163 - WEGNER P.

Machine organisation for multiprogramming

ACM 22nd Nat. Conf. - 1967 - pp. 135 - 150

164 - WEILL J.W.

A heuristic for page turning in a multiprogrammed computer

Comm. ACM - vol. 5 n° 9 - Sept. 1962 - pp. 480 - 481

165 - WILKES M.V.

Slave memories and dynamic storage allocation

IEEE Tr. - vol. EC 14 n° 4 - April 1965 - pp. 270 - 271

166 - WILKES M.V.

The design of multi-access computer systems

Part 1 : The computer Journal - vol. 10 n° 1 - pp. 1 - 9

May 1967

Part 2 : The computer Journal - vol. 10 n° 4 - pp. 315 - 320

Feb. 1968

167 - WILKES M.V.

The growth of interest in microprogramming

Comp. Sur. - 1969 - vol. 1 n° 3 - pp. 139 - 145

168 - WIRTH N.

On multiprogramming, machine coding and computer organisation

Comm. ACM - vol. 12 n° 9 - Sept. 1969 - pp. 489 - 498

169 - WITT B.I.

Dynamic storage allocation for a time shared system

IBM Sys. J. - vol. 2 - Sept. - Dec. 1963 - pp. 230 - 239

II - CLASSEMENT INDEXE

A		C	
ABERTH O.	1	CAMPEAU J.O.	26
ANDERSON J.P.	2	CANTRELL E.	27
ANDERSON D.W.	3	CHANG H.Y.	28
ANDERSON S.F.	4	CHANG W.A.	29
ARDEN B.W.	5	CHEN E.C.Y.	30
ARSAC J.	6	CHIN TUNG	31
ASCHENBRENNER R.A.	7	CHU W.W.	32
ASCHER J.R.	8	CODD E.F.	33
		COFFMANN E.F.	34, 35, 36
		COMEAU L.W.	37
		COMFORT W.T.	38, 39
		CONTI C.J.	40
		CONWAY M.E.	41
		CORBATO F.J.	42
		COTTON L.W.	43, 44
		COULEUR J.F.	45
		CRITCHLOW A.J.	46
		CURTIN W.A.	47
B		D	
BAER J.L.	9		
BALDWIN F.R.	10		
BARNES G.H.	11		
BAUER W.F.	12		
BELADY L.A.	13		
BERNSTEIN A.J.	14		
BEUSCHER H.J.	15		
BIBB J.	16, 17		
BOLAND L.J.	18		
BORCK W.C.	19, 20		
BOVETT D.P.	9	DAHM D.M.	48
BRAWN B.	21	DALEY R.C.	49
BROWN J.L.	22	DEBRAINE P.	50
BROWN R.M.	11	DELLER G.T.	51
BRYAN G.I.	23	DENNING P.J.	52, 53
BURNETT G.J.	24, 25	DENNIS J.B.	49, 54, 55

DIJKSTRA E.W. 56
DOODY D.T. 22
DRUMMOND H.E. 57

E

EARLE J.G. 4
EDWARDS D.G. 58
ELROD T.H. 59
ENGLAND H.W. 60
ERICKSON G.A. 61
ESTRIN G. 16, 17

GIBSON C.T. 75
GIBSON D.H. 40, 76
GIBSON W.B. 10
GLASER E.L. 45
GOLDSCHMIDT R.E. 4
GRAHAM R.M. 42, 77
GRAHAM W.R. 78
GRANITO G.D. 18
GREEN J. 79
GREGORY J.G. 19
GUPTA S. 80, 81
GUSTAVSON F. 21

H

F

FINE G.H. 62
FLORES I. 63
FLYNN M.J. 7, 64, 65, 66
FOSTER C.C. 67, 68
FOTHERINGHAM J. 69
FRIEDMAN T.D. 70
FUCHEL K. 71
FUGG P. 22
FULLER R.H. 72

G

GALLER B.A. 5
GARNER H.L. 73
GECSEI J. 74
GERBSTADT F.H. 48

HABERMAN A.N. 82
HALEY G. 30
HANLON A.G. 83
HEBENSTREIT J. 84
HELLER S. 71
HELLERMAN L. 85, 86, 87
HENNIE F.C. 88
HOERNES C.E. 87
HOKOM R.A. 25
HOLLANDER G.L. 89
HOLT A.W. 90
HORWITZ L.P. 91
HOWARTH D.J. 92
HUDSON J.R. 19
HUMPHREY T.A. 93
HUSSON S.S. 94

I

ILIFFE J.K.	95
IRONS E.T.	96
IVERSON K.E.	97

J

JACKSON C.W.	62
JODEIT J.G.	95
JOHNSON R.R.	98
JONES P.D.	92

K

KARP R.M.	91
KATO M.	11
KEEFE D.D.	99
KILBURN T.	58
KINNIMENT D.J.	100
KLEINROCK L.	101
KNOWLES H.E.	100
KOCZELA L.J.	25, 102, 103
KOHAVI Z.	104
KUCK D.J.	11, 105
KUENHER C.J.	106, 107

L

LAMPSON B.W.	108
LANIGAN M.J.	58

LASSER D.J.	109
LAVALLEE P.	104
LAVER H.C.	110
LAVINGTON S.H.	100
LAWSON H.W.	111
LEE F.S.	112
LEHMAN M.M.	113, 114, 115
LINCOLN N.R.	107
LINDQUIST A.B.	37, 116
LIPTAY J.S.	117
LITWIN S.	118
LOW P.R.	64

M

MADNICK S.E.	119
MAHER R.S.	120
MANNING E.	121
MARCOTTE A.U.	18
MATTSON R.L.	74
Mc CULLOUGH J.D.	122
Mc DOUGALL M.H.	123
Mc GEE W.C.	124
Mc ISAAC P.V.	62
Mc NAUGHTON P.C.	112
Mc REYNOLDS R.C.	20
MEADE R.M.	125
MENDELSON R.M.	60
MESSINA B.U.	18
MILLER R.E.	91
MINSKY M.L.	126
MURPHEY J.O.	127
MURTHA J.C.	19, 128
MUTSCHLER E.G.	61

N - O

NATARAJAN N.K.	129
O'BRIEN T.C.	5
OLIVER G.A.	45
O'NEILL R.W.	130

P

PACELLI M.M.	48
PADEGS A.	131
PICKERING G.E.	61
PITKOWSKI S.H.	40
POLAND C.B.	10
POWERS D.M.	4
PRYWES N.S.	118
PYKE T.N.	132

R

RAMAMOORTHY C.V.	134, 135, 136
RANDELL B.	106, 107, 133
REITER R.	137
ROBINSON G.A.	7
RODER J.	138
ROSENE A.F.	138, 139
ROSENFELD J.L.	114, 140
ROSIN R.F.	141, 142
RUSSEL B.	16, 17

S

SAYRE D.	143
SCANLON J.M.	28
SCHLAEPPI H.P.	144
SCHON H.	145
SCHWARTZ J.	146
SCHWARTZ J.I.	34
SEAMAN P.H.	147
SEEBER R.R.	37, 116
SENZIG D.N.	148
SHAPIRO E.	166
SHEDLER G.S.	119
SHEMER J.E.	80, 81, 149
SHERLOCK J.F.	150
SHIPPEY G.A.	149
SKINNER C.E.	8
SLOTNICK D.L.	11, 20
SLUTZ D.R.	74
SMITH J.L.	151
SMITH J.W.	18
SOUCY R.C.	147
SPARACIO F.J.	3
SPEIERMAN K.H.	122
SQUIRE J.S.	73
STABLER E.P.	152
STANGA D.C.	153
STEVENS D.F.	154
STOKES R.A.	11
STONE H.S.	155, 156
SUMNER F.H.	30, 58

T

THOMAS P.A.V.	129	WIRTH N.	168
THORLIN J.F.	157	WITT B.I.	169
THORNTON J.E.	158	WYLD M.T.	92
TJADEN G.J.	65	WYLES H.	24
TOMASULO R.M.	3, 159		
TOY W.N.	15		
TRAIGER I.L.	74	Y - Z	
TSUJIGADO M.	160		
TSYCHIYA M.	136	YANG S.C.	70
TUCKER A.B.	66	ZURCHER F.W.	122
TUCKER S.G.	161, 162		
TURN R.	16		

V

VAN HORN E.C.	55
VARIAN L.C.	36
VYSSOTSKY V.A.	42

W

WADE R.M.	127
WANG G.Y.	102
WEGNER P.	163
WEILL J.W.	164
WEISMAN C.	34
WESTERVELT F.H.	5
WILKES M.V.	165, 166, 167
WINOGRAD S.	91

III - CLASSEMENT MATIERES

- Généralités

6, 12, 39, 47, 59, 60, 70, 88, 89, 126

- Architecture générale

1, 3, 7, 11, 19, 20, 22, 25, 26, 30, 38, 40, 46, 56, 64, 71,
78, 80, 103, 112, 118, 122, 127, 144, 145, 146, 153, 163, 166,
168

- Structures parallèles ou multiples

14, 16, 17, 41, 65, 68, 72, 78, 102, 113, 116, 128

- Temps partagé

23, 29, 34, 35, 45, 75, 85, 132, 169

- Techniques d'adressage, pagination, segmentation

5, 13, 21, 31, 36, 49, 50, 52, 54, 55, 58, 62, 63, 67, 69, 76,
81, 83, 87, 90, 91, 95, 100, 101, 106, 107, 120, 143, 151, 155,
164, 165

- Mémoire

18, 37, 53, 74, 77, 93, 105, 110, 117, 129, 133, 138, 139, 156

- Microprogrammation

66, 79, 84, 94, 125, 136, 142, 152, 162, 167

- Arithmétique, instructions

4, 73, 131, 157, 158, 159

- Contrôle

15, 28, 77, 82, 87, 104, 121, 134

- Communications

43, 44

- Performances

8, 24, 27, 57, 98, 99, 109, 114, 130, 148, 149

- Simulation, émulation

79, 129, 150, 161

- Programmation, multiprogrammation

2, 9, 10, 32, 33, 42, 48, 51, 61, 79, 82, 86, 92, 96, 97, 108,
111, 115, 119, 124, 135, 137, 140, 141, 147, 154, 160

TABLE DES MATIERES

	page
CHAPITRE I - INTRODUCTION	1
CHAPITRE II - INTERCONNEXION ET CADENCEMENT DU SYSTEME	9
2.1 - Systèmes d'interconnexion	10
2.2 - Distribution du signal	18
2.3 - Cadencement du système	19
CHAPITRE III - DESCRIPTION DU SYSTEME	23
3.1 - Caractéristiques opérationnelles	24
3.2 - Composants du système central	26
3.3 - Unité d'exécution	27
3.4 - Mémoire générale	28
3.5 - Mémoire auxiliaire	28
3.6 - Système d'entrée-sortie	28
3.7 - Unité de contrôle centrale, unité de contrôle périphérique	33
3.8 - Résumé des caractéristiques du système	34
3.9 - Configuration	36
CHAPITRE IV - MECANISME D'ADRESSAGE, MEMOIRE PRINCIPALE, MEMOIRE LOCALE, MEMOIRE ETENDUE	37
4.1 - Adressage	37
4.2 - Mémoire principale	43
4.3 - Mémoire locale	49
4.4 - Mémoire étendue	57

CHAPITRE V	- UNITE DE COMMANDE, DE TRAVAIL, D'EXECUTION	59
5.1	- Unité de commande	59
5.2	- Unité locale de travail	62
5.3	- Unité d'exécution	67
CHAPITRE VI	- MACHINES SEQUENTIELLES ET MICROPROGRAMMATION	73
6.1	- Machines séquentielles et signaux d'état	73
6.2	- Microprogrammation	78
CHAPITRE VII	- UNITES PERIPHERIQUES ET DE CONTROLE	83
7.1	- Unités périphériques	83
7.2	- Unité de liaison canal	85
7.3	- Problème de transfert	87
7.4	- Unités de contrôle	90
CHAPITRE VIII	- SYSTEME DE COMMANDE ET PROGRAMMATION	93
8.1	- Système de commande	93
8.2	- Programmation	97
CHAPITRE IX	- CONCLUSION	103
BIBLIOGRAPHIE		105
TABLE DES MATIERES		135

